EAST POINT COLLEGE OF ENGINEERING & TECHNOLOGY

'Jnana Prabha', Virgo Nagar Post, Bengaluru-560049

Department of Computer Science and Engineering

Academic Year: 2024-25

LABORATORY MANUAL

Semester : VI
Subject : Machine Learning lab
Subject Code : BCSL606

NAME: _____

USN: _____

SECTION: _____

PROGRAM OUTCOMES

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modernengineering and IT tools including prediction and modeling to complex engineering activitieswith an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the Engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life -long learning: Recognize the need for and have the preparation and ability to

engage in independent and life -long learning in the broadest context of technological change.

EAST POINT COLLEGE OF ENGINEERING & TECHNOLOGY

Department of Computer Science and Engineering

INSTITUTE VISION AND MISSION

VISION

The East Point College of Engineering and Technology aspires to be a globally acclaimed institution, recognized for excellence in engineering education, applied research and nurturing students for holistic development.

MISSION

M1: To create engineering graduates through quality education and to nurtureinnovation, creativity and excellence in teaching, learning and research

M2: To serve the technical, scientific, economic and societal developmental needsof our communities

M3: To induce integrity, teamwork, critical thinking, personality development andethics in students and to lay the foundation for lifelong learning

EAST POINT COLLEGE OF ENGINEERING & TECHNOLOGY

Department of Computer Science and Engineering

DEPARTMENT VISION AND MISSION

VISION

The department aspires to be a Centre of excellence in Computer Science & Engineering to develop competent professionals through holistic development.

MISSION

M1: To create successful Computer Science Engineering graduates through effective pedagogies, the latest tools and technologies, and excellence in teaching and learning.

M2: To augment experiential learning skills to serve technical, scientific, economic, and social developmental needs.

M3: To instil integrity, critical thinking, personality development, and ethics in students for a successful career in Industries, Research, and Entrepreneurship.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO 1: To produce graduates who can perform technical roles to contribute effectively in software industries and R&D Centre.

PEO 2: To produce graduates having the ability to adapt and contribute in key domains of computer science and engineering to develop competent solutions.

PEO 3: To produce graduates who can provide socially and ethically responsible solutions while adapting to new trends in the domain to carve a successful career in the industry.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: To conceptualize, model, design, simulate, analyse, develop, test, and validate computing systems and solve technical problems arising in the field of computer science & engineering.

PSO2: To specialize in the sub-areas of computer science & engineering systems such as cloud computing, Robotic Process Automation, cyber security, big data analytics, user interface design, and IOT to meet industry requirements.

PSO3: To build innovative solutions to meet the demands of the industry using appropriate tools and techniques.

COURSE LEARNING OBJECTIVES

- To become familiar with data and visualize univariate, bivariate, and multivariate data using statistical techniques and dimensionality reduction.
- To understand various machine learning algorithms such as similarity-based learning, regression, decision trees, and clustering.
- To familiarize with learning theories, probability-based models and developing the skills required for decision-making in dynamic environments.

COURSE OUTCOMES

At the end of the course the student will be able to:

CO1: Illustrate the principles of multivariate data and apply dimensionality reduction techniques.

CO2: Demonstrate similarity-based learning methods and perform regression analysis.

CO3: Develop decision trees for classification and regression problems, and Bayesian models for probabilistic learning.

CO4: Implement the clustering algorithms to share computing resources.

Machine Learning lab		Semester	6
Course Code	**BCSL606**	**CIE Marks**	**50**
Teaching Hours/Week (L:T:P: S)	**0:0:2:0**	**SEE Marks**	**50**
Credits	**01**	**Exam Hours**	**2**
Examination type (SEE)	**Practical**		

Sl. No	Experiments
1	Develop a program to create histograms for all numerical features and analyze the distribution of each feature. Generate box plots for all numerical features and identify any outliers. Use California Housing dataset.
2	Develop a program to Compute the correlation matrix to understand the relationships between pairs of features. Visualize the correlation matrix using a heatmap to know which variables have strong positive/negative correlations. Create a pair plot to visualize pairwise relationships between features. Use California Housing dataset.
3	Develop a program to implement Principal Component Analysis (PCA) for reducing the dimensionality of the Iris dataset from 4 features to 2.
4	For a given set of training data examples stored in a .CSV file, implement and demonstrate the Find-S algorithm to output a description of the set of all hypotheses consistent with the training examples.
5	Develop a program to implement k-Nearest Neighbour algorithm to classify the randomly generated 100 values of *x* in the range of [0,1]. Perform the following based on dataset generated. a. Label the first 50 points {*x1*,……,*x50*} as follows: if (xi ≤ 0.5), then *xi* ε Class1, else *xi* ε Class1 b. Classify the remaining points, *x51*,……,*x100* using KNN. Perform this for *k=1,2,3,4,5,20,30*
6	Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.
7	Develop a program to demonstrate the working of Linear Regression and Polynomial Regression. Use Boston Housing Dataset for Linear Regression and Auto MPG Dataset (for vehicle fuel efficiency prediction) for Polynomial Regression.
8	Develop a program to demonstrate the working of the decision tree algorithm. Use Breast Cancer Data set for building the decision tree and apply this knowledge to classify a new sample.
9	Develop a program to implement the Naive Bayesian classifier considering Olivetti Face Data set for training. Compute the accuracy of the classifier, considering a few test data sets.
10	Develop a program to implement k-means clustering using Wisconsin Breast Cancer data set and visualize the clustering result.

Assessment Details (both CIE and SEE):

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%.

The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50) and for the SEE minimum passing mark is 35% of the maximum marks (18 out of 50 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/

course if the student secures a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together.

Continuous Internal Evaluation (CIE): CIE marks for the practical course are 50 Marks.

- The split-up of CIE marks for record/ journal and test are in the ratio 60:40.
- Each experiment is to be evaluated for conduction with an observation sheet and record write-up. Rubrics for the evaluation of the journal/write-up for hardware/software experiments are designed by the faculty who is handling the laboratory session and are made known to students at the beginning of the practical session.
- Record should contain all the specified experiments in the syllabus and each experiment write-up will be evaluated for 10 marks.
- Total marks scored by the students are scaled down to 30 marks (60% of maximum marks).
- Weightage to be given for neatness and submission of record/write-up on time.
- Department shall conduct a test of 100 marks after the completion of all the experiments listed in the syllabus.
- In a test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weightage of 60% and the rest 40% for viva-voce.
- The suitable rubrics can be designed to evaluate each student's performance and learning ability.
- The marks scored shall be scaled down to 20 marks (40% of the maximum marks).

The Sum of scaled-down marks scored in the report write-up/journal and marks of a test is the total CIE marks scored by the student.

Semester End Evaluation (SEE):

SEE marks for the practical course are 50 Marks.

SEE shall be conducted jointly by the two examiners of the same institute, examiners are appointed by the Head of the Institute.

The examination schedule and names of examiners are informed to the university before the conduction of the examination. These practical examinations are to be conducted between the schedule mentioned in the academic calendar of the University.

All laboratory experiments are to be included for practical examination.

(Rubrics) Breakup of marks and the instructions printed on the cover page of the answer script to be strictly adhered to by the examiners. **OR** based on the course requirement evaluation rubrics shall be decided jointly by examiners.

Students can pick one question (experiment) from the questions lot prepared by the examiners jointly.

Evaluation of test write-up/ conduction procedure and result/viva will be conducted jointly by examiners.

General rubrics suggested for SEE are mentioned here, writeup-20%, Conduction procedure and result in -60%, Viva-voce 20% of maximum marks. SEE for practical shall be evaluated for 100 marks and scored marks shall be scaled down to 50 marks (however, based on course type, rubrics shall be decided by the examiners)

Change of experiment is allowed only once and 15% of Marks allotted to the procedure part are to be made zero.

The minimum duration of SEE is 02 hours

Suggested Learning Resources:

Books:

1. S Sridhar and M Vijayalakshmi, “Machine Learning”, Oxford University Press, 2021.
2. M N Murty and Ananthanarayana V S, “Machine Learning: Theory and Practice”, Universities Press (India) Pvt. Limited, 2024.

Web links and Video Lectures (e-Resources):

- https://www.drssridhar.com/?page_id=1053
- https://www.universitiespress.com/resources?id=9789393330697
- https://onlinecourses.nptel.ac.in/noc23_cs18/preview

Index					
Sl. No.	**Program List**	**CO**	**PO**	**RBT**	**Page No.**
1	Develop a program to create histograms for all numerical features and analyze the distribution of each feature. Generate box plots for all numerical features and identify any outliers. Use California Housing dataset.	CO1	PO1, PO2, PO3, PO5, PSO1,2,3	L3	16
2	Develop a program to Compute the correlation matrix to understand the relationships between pairs of features. Visualize the correlation matrix using a heatmap to know which variables have strong positive/negative correlations. Create a pair plot to visualize pairwise relationships between features. Use California Housing dataset.	CO1	PO1, PO2, PO3, PO5, PSO1,2,3	L3	20
3	Develop a program to implement Principal Component Analysis (PCA) for reducing the dimensionality of the Iris dataset from 4 features to 2.	CO1	PO1, PO2, PO3, PO5, PSO1,2,3	L3	23
4	For a given set of training data examples stored in a .CSV file, implement and demonstrate the Find-S algorithm to output a description of the set of all hypotheses consistent with the training examples.	CO1	PO1, PO2, PO3, PO5, PSO1,2,3	L3	25
5	Develop a program to implement k-Nearest Neighbour algorithm to classify the randomly generated 100 values of *x* in the range of [0,1]. Perform the following based on dataset generated. a. Label the first 50 points {*x1*,……,*x50*} as follows: if (xi ≤ 0.5), then *xi* ε Class1, else *xi* ε Class1 b. Classify the remaining points, *x51*,……,*x100* using KNN. Perform this for *k=1,2,3,4,5,20,30*	CO2	PO1, PO2, PO3, PO5, PSO1,2,3	L3	27
6	Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.	CO2	PO1, PO2, PO3, PO5, PSO1,2,3	L3	42
7	Develop a program to demonstrate the working of Linear Regression and Polynomial Regression. Use Boston Housing Dataset for Linear Regression and Auto MPG Dataset (for vehicle fuel efficiency prediction) for Polynomial Regression.	CO2	PO1, PO2, PO3, PO5, PSO1,2,3	L3	44
8	Develop a program to demonstrate the working of the decision tree algorithm. Use Breast Cancer Data set for building the decision tree and apply this knowledge to classify a new sample.	CO3	PO1, PO2, PO3, PO5, PSO1,2,3	L3	48

9	Develop a program to implement the Naive Bayesian classifier considering Olivetti Face Data set for training. Compute the accuracy of the classifier, considering a few test data sets.	CO3	PO1, PO2, PO3, PO5, PSO1,2,3	L3	50
10	Develop a program to implement k-means clustering using Wisconsin Breast Cancer data set and visualize the clustering result.	CO4	PO1, PO2, PO3, PO5, PSO1,2,3	L3	54
Viva Questions and Answers					58

3. Course Articulation Matrix

Cos	**Pos**												**PSOs**		
	PO 1	**PO 2**	**PO 3**	**PO 4**	**PO 5**	**PO 6**	**PO 7**	**PO 8**	**PO 9**	**PO 10**	**PO 11**	**PO 12**	**PSO 1**	**PSO 2**	**PSO 3**
CO1	3	3	3	2	3	2	-	-	2	-	-	2	3	3	3
CO2	3	3	3	2	3	2	-	-	2	-	-	2	3	3	3
CO3	3	3	3	2	3	2	-	-	2	-	-	2	3	3	3
CO4	3	3	3	2	3	2	-	-	2	-	-	2	3	3	3

3 - High Correlation 2 - Medium Correlation 1 – Low Correlation

Introduction to Machine Learning Lab

What is Machine Learning?

Machine Learning (ML) is a subfield of artificial intelligence (AI) that focuses on the development of algorithms that can learn from and make predictions or decisions based on data. Unlike traditional software programs, which follow explicit instructions coded by humans, machine learning models automatically improve their performance as they are exposed to more data.

In this **Machine Learning Lab**, students will explore and implement fundamental machine learning algorithms, focusing on understanding both the mathematical foundations and practical applications. You will work with various datasets to apply and experiment with techniques used in supervised and unsupervised learning, regression, classification, clustering, and dimensionality reduction.

Tools and Environment

Programming Language: Python

Python is one of the most widely used programming languages in the field of data science and machine learning. It offers extensive libraries and frameworks for implementing machine learning algorithms, including:

- **NumPy** for numerical computing
- **Pandas** for data manipulation and analysis
- **Matplotlib** and **Seaborn** for data visualization
- **Scikit-learn** for machine learning algorithms and utilities
- **TensorFlow** and **Keras** for deep learning models

Jupyter Notebooks

Jupyter Notebooks provide an interactive environment for writing and executing Python code. It allows you to mix code with markdown text and visualizations, making it an excellent tool for experimenting with machine learning algorithms and visualizing results.

Installation Procedure:

Step 1: Install Python

1. Go to the official Python website: https://www.python.org/downloads/.
2. Download the appropriate version for your operating system (Windows, macOS, or Linux).
3. Run the installer and ensure you check the box that says **Add Python to PATH** before clicking on **Install Now**.

4. Once the installation is complete, open a terminal or command prompt and type `python --version` to verify that Python is installed correctly.

Step 2: Install pip (Python Package Installer)

pip is the package manager for Python, and it should already be installed with Python (since version 3.4). To verify, open a terminal or command prompt and type:

```
pip --version
```

If pip is installed, you'll see the version number. If not, you have to install pip.

Step 3: Install Jupyter Notebook using pip

Once Python and pip are set up, you can install **Jupyter Notebook** using the following command in your terminal or command prompt:

```
pip install notebook
```

This will download and install Jupyter and its dependencies.

Step 4: Launch Jupyter Notebook

After the installation is complete, you can launch Jupyter Notebook from the command line by typing:

```
jupyter notebook
```

This will open a new tab in your default web browser with the Jupyter Notebook interface, allowing you to create and run Python notebooks.

Step 5: Optional - Install Jupyter Notebook with Anaconda

Alternatively, you can install **Anaconda**, which is a Python distribution that comes with Jupyter Notebook and other useful data science libraries pre-installed. Here's how:

1. Download the Anaconda installer from the official website: https://www.anaconda.com/products/distribution
2. Run the installer and follow the instructions.
3. Once installed, open the Anaconda Navigator application and click on **Launch** under Jupyter Notebook.

Step 6: Optional - Install Additional Libraries

Inside Jupyter Notebook, you may install additional Python libraries, such as **NumPy**, **Pandas**, **Matplotlib**, or **Scikit-learn** for machine learning. You can install these libraries using pip as follows:

```
pip install numpy pandas matplotlib scikit-learn
```

Datasets

California Housing Dataset

The **California Housing dataset** is a classic regression problem dataset consisting of information about the housing prices in California. It includes numerical features such as the number of rooms, population, and median income for each block in California. This dataset is used to perform tasks like regression analysis and visualize distributions.

Iris Dataset

The **Iris dataset** is a famous classification dataset that contains data about three species of Iris flowers, each represented by four numerical features: sepal length, sepal width, petal length, and petal width. This dataset is widely used for classification tasks.

Boston Housing Dataset

The **Boston Housing dataset** consists of information about housing prices in Boston. It includes features such as crime rate, property tax rate, and average number of rooms in a neighborhood. It is often used for linear regression tasks.

Olivetti Face Dataset

The **Olivetti Face dataset** contains images of 40 different subjects, each having multiple images under different lighting conditions and facial expressions. This dataset is used for facial recognition and classification tasks.

Wisconsin Breast Cancer Dataset

The **Wisconsin Breast Cancer dataset** contains features derived from a digitized image of a breast cancer tumor, which can be used for classification into benign or malignant categories. It's commonly used for testing classification algorithms.

Algorithms Covered

1. k-Nearest Neighbors (KNN)

The **k-Nearest Neighbors (KNN)** algorithm is a non-parametric, lazy learning algorithm used for classification and regression tasks. Given a set of training data points, it classifies a new point by finding the **k** closest points (neighbors) to it and assigning the most common class label (for classification) or the average value (for regression).

- **Applications**: Classification (e.g., handwritten digit recognition) and regression (e.g., predicting house prices).

- **Strengths**: Simple to implement, intuitive.
- **Limitations**: Computationally expensive with large datasets, sensitive to the choice of **k**.

2. Principal Component Analysis (PCA)

PCA is a dimensionality reduction technique that transforms data into a new coordinate system, where the first few coordinates (principal components) retain most of the variation in the data. It is commonly used to reduce the number of features while retaining essential patterns.

- **Applications**: Image compression, feature selection.
- **Strengths**: Reduces overfitting, improves performance for models that suffer from high dimensionality.
- **Limitations**: Can be sensitive to scaling, may not work well if features have nonlinear relationships.

3. Locally Weighted Regression (LWR)

Locally Weighted Regression is a non-parametric regression method that fits a regression model locally, by considering only the nearest points to the target value. The model is adjusted for each query point based on its local neighborhood.

- **Applications**: Non-linear regression problems, time-series forecasting.
- **Strengths**: Can capture non-linear relationships.
- **Limitations**: Requires more computation as it is instance-based.

4. Linear Regression

Linear Regression is a statistical method for modeling the relationship between a dependent variable and one or more independent variables. It assumes that there is a linear relationship between the input features and the output.

- **Applications**: Predicting continuous variables, such as housing prices or stock prices.
- **Strengths**: Simple to understand and implement, easy to interpret.
- **Limitations**: Assumes a linear relationship, sensitive to outliers.

5. Polynomial Regression

Polynomial Regression is an extension of linear regression that models the relationship between the independent variable and the dependent variable as an **n-th degree polynomial**.

- **Applications**: Modeling non-linear relationships.
- **Strengths**: Can capture complex relationships.
- **Limitations**: Can lead to overfitting if the degree of the polynomial is too high.

6. Decision Trees

A **Decision Tree** is a flowchart-like structure where each internal node represents a decision based on a feature, and each leaf node represents an outcome. Decision Trees can be used for both classification and regression tasks.

- **Applications**: Classifying whether a customer will buy a product, predicting medical diagnoses.
- **Strengths**: Easy to interpret, no need for feature scaling.
- **Limitations**: Prone to overfitting, can be unstable with small data changes.

7. Naive Bayes

Naive Bayes is a probabilistic classifier based on Bayes' Theorem, assuming that the features are conditionally independent. It is commonly used for classification tasks.

- **Applications**: Text classification, spam detection, sentiment analysis.
- **Strengths**: Simple, efficient, works well with small datasets.
- **Limitations**: Assumes independence among features, which might not always hold true.

8. K-Means Clustering

K-Means is an unsupervised clustering algorithm that partitions data into **k** clusters based on the distance between data points and centroids. It aims to minimize the variance within each cluster.

- **Applications**: Market segmentation, image compression, anomaly detection.
- **Strengths**: Simple and fast, scalable.
- **Limitations**: Sensitive to the choice of **k**, assumes spherical clusters.

9. Find-S Algorithm

Find-S is a simple machine learning algorithm used for concept learning. It finds the most specific hypothesis that is consistent with all positive examples in the training data.

- **Applications**: Used in simple concept learning tasks, such as classification based on rules.
- **Strengths**: Easy to understand and implement.
- **Limitations**: Only works for learning from positive examples.

EXPERIMENT-1

Develop a program to create histograms for all numerical features and analyze the distribution of each feature. Generate box plots for all numerical features and identify any outliers. Use California Housing dataset.

About the Experiment:

The California Housing dataset, which is commonly used in data science for regression tasks. The dataset contains information about different houses in California, such as their median house value, average rooms, average bedrooms, population, households, and other features. The goal of this task is to analyze the distribution of numerical features in the dataset and identify potential outliers using histograms and box plots.

Key Concepts are:

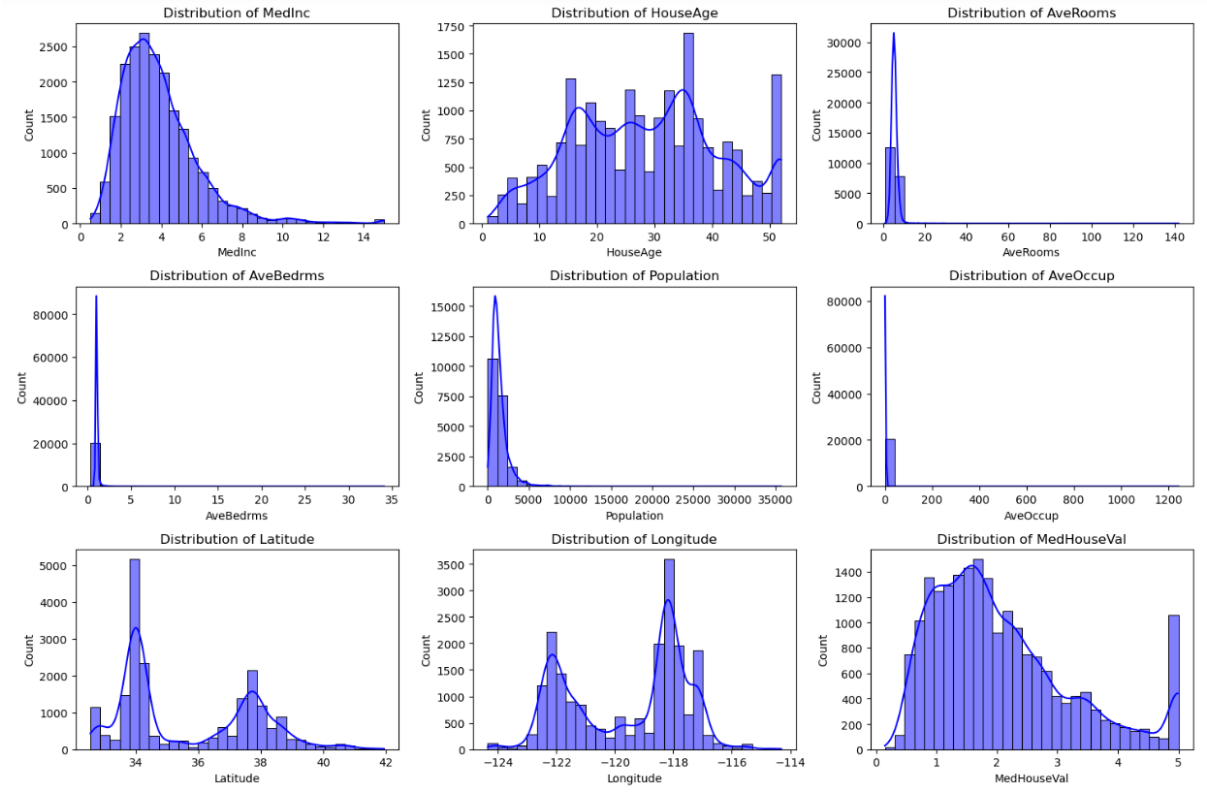
- **Histograms**: A histogram is a graphical representation of the distribution of a dataset. It shows the frequency of data points within specified intervals or bins. This helps to analyze the distribution (normal, skewed, bimodal, etc.) of numerical features.
- **Box Plots**: A box plot provides a graphical summary of the distribution of numerical data through its quartiles. It highlights the median, 25th and 75th percentiles, and potential outliers (data points that lie outside the "whiskers" of the box plot).
- **Outliers**: Outliers are data points that differ significantly from the majority of the data. They can arise due to measurement errors or may represent important but rare occurrences. Identifying outliers is crucial because they may impact the performance of machine learning models.

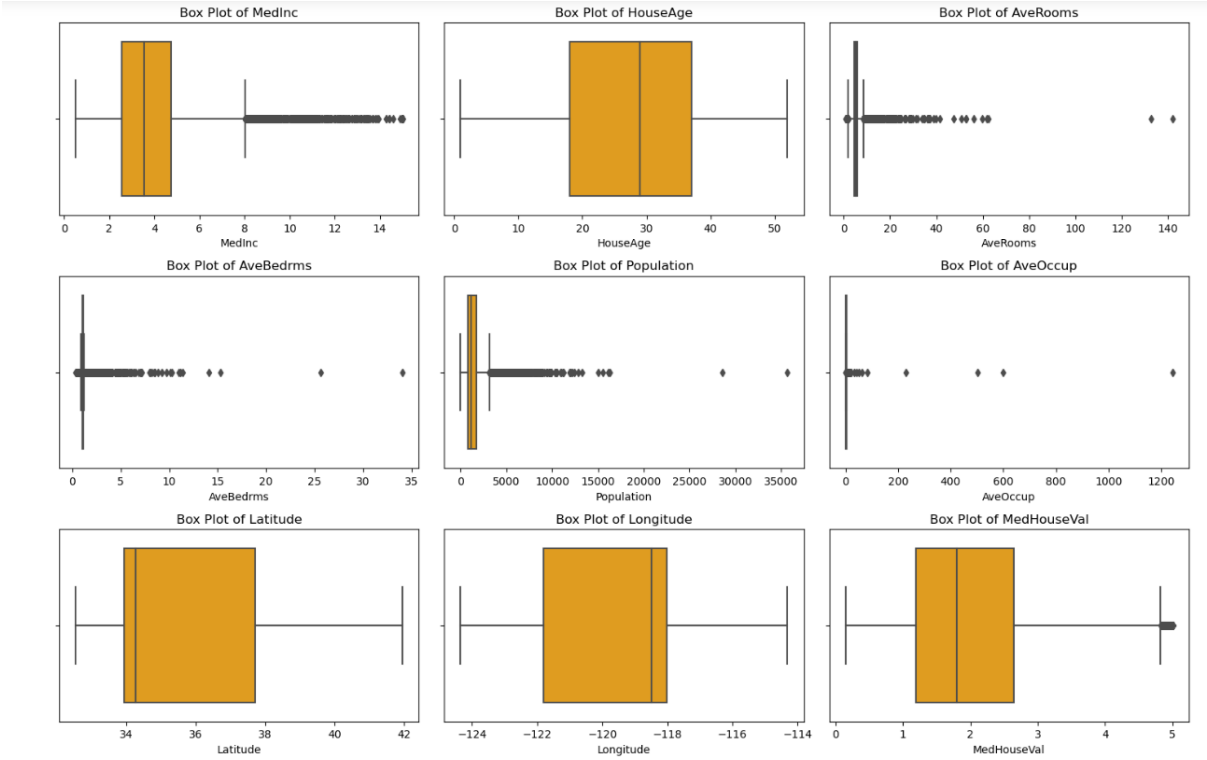
Program:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing

# Step 1: Load the California Housing dataset
data = fetch_california_housing(as_frame=True)
housing_df = data.frame

# Step 2: Create histograms for numerical features
numerical_features =  housing_df.select_dtypes(include=[np.number]).columns

# Plot histograms
plt.figure(figsize=(15, 10))
for i, feature in enumerate(numerical_features):
```

```
    plt.subplot(3, 3, i + 1)
    sns.histplot(housing_df[feature], kde=True, bins=30, color='blue')
    plt.title(f'Distribution of {feature}')
plt.tight_layout()
plt.show()

# Step 3: Generate box plots for numerical features
plt.figure(figsize=(15, 10))
for i, feature in enumerate(numerical_features):
    plt.subplot(3, 3, i + 1)
    sns.boxplot(x=housing_df[feature], color='orange')
    plt.title(f'Box Plot of {feature}')
plt.tight_layout()
plt.show()

# Step 4: Identify outliers using the IQR method
print("Outliers Detection:")
outliers_summary = {}
for feature in numerical_features:
    Q1 = housing_df[feature].quantile(0.25)
    Q3 = housing_df[feature].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = housing_df[(housing_df[feature] < lower_bound) |
                          (housing_df[feature] > upper_bound)]
    outliers_summary[feature] = len(outliers)
    print(f"{feature}: {len(outliers)} outliers")

# Optional: Print a summary of the dataset
print("\nDataset Summary:")
print(housing_df.describe())
```

Output:


Distribution of MedInc
Count
MedInc
Distribution of HouseAge
Count
HouseAge
Distribution of AveRooms
Count
AveRooms
Distribution of AveBedrms
Count
AveBedrms
Distribution of Population
Count
Population
Distribution of AveOccup
Count
AveOccup
Distribution of Latitude
Count
Latitude
Distribution of Longitude
Count
Longitude
Distribution of MedHouseVal
Count
MedHouseVal


Box Plot of MedInc
MedInc
Box Plot of HouseAge
HouseAge
Box Plot of AveRooms
AveRooms
Box Plot of AveBedrms
AveBedrms
Box Plot of Population
Population
Box Plot of AveOccup
AveOccup
Box Plot of Latitude
Latitude
Box Plot of Longitude
Longitude
Box Plot of MedHouseVal
MedHouseVal

```
Outliers Detection:
MedInc: 681 outliers
HouseAge: 0 outliers
AveRooms: 511 outliers
AveBedrms: 1424 outliers
Population: 1196 outliers
AveOccup: 711 outliers
Latitude: 0 outliers
Longitude: 0 outliers
MedHouseVal: 1071 outliers

Dataset Summary:
              MedInc      HouseAge      AveRooms     AveBedrms    Population  \
count  20640.000000  20640.000000  20640.000000  20640.000000  20640.000000
mean       3.870671     28.639486      5.429000      1.096675   1425.476744
std        1.899822     12.585558      2.474173      0.473911   1132.462122
min        0.499900      1.000000      0.846154      0.333333      3.000000
25%        2.563400     18.000000      4.440716      1.006079    787.000000
50%        3.534800     29.000000      5.229129      1.048780   1166.000000
75%        4.743250     37.000000      6.052381      1.099526   1725.000000
max       15.000100     52.000000    141.909091     34.066667  35682.000000

            AveOccup      Latitude     Longitude   MedHouseVal
count  20640.000000  20640.000000  20640.000000  20640.000000
mean       3.070655     35.631861   -119.569704      2.068558
std       10.386050      2.135952      2.003532      1.153956
min        0.692308     32.540000   -124.350000      0.149990
25%        2.429741     33.930000   -121.800000      1.196000
50%        2.818116     34.260000   -118.490000      1.797000
75%        3.282261     37.710000   -118.010000      2.647250
max     1243.333333     41.950000   -114.310000      5.000010
```

EXPERIMENT-2

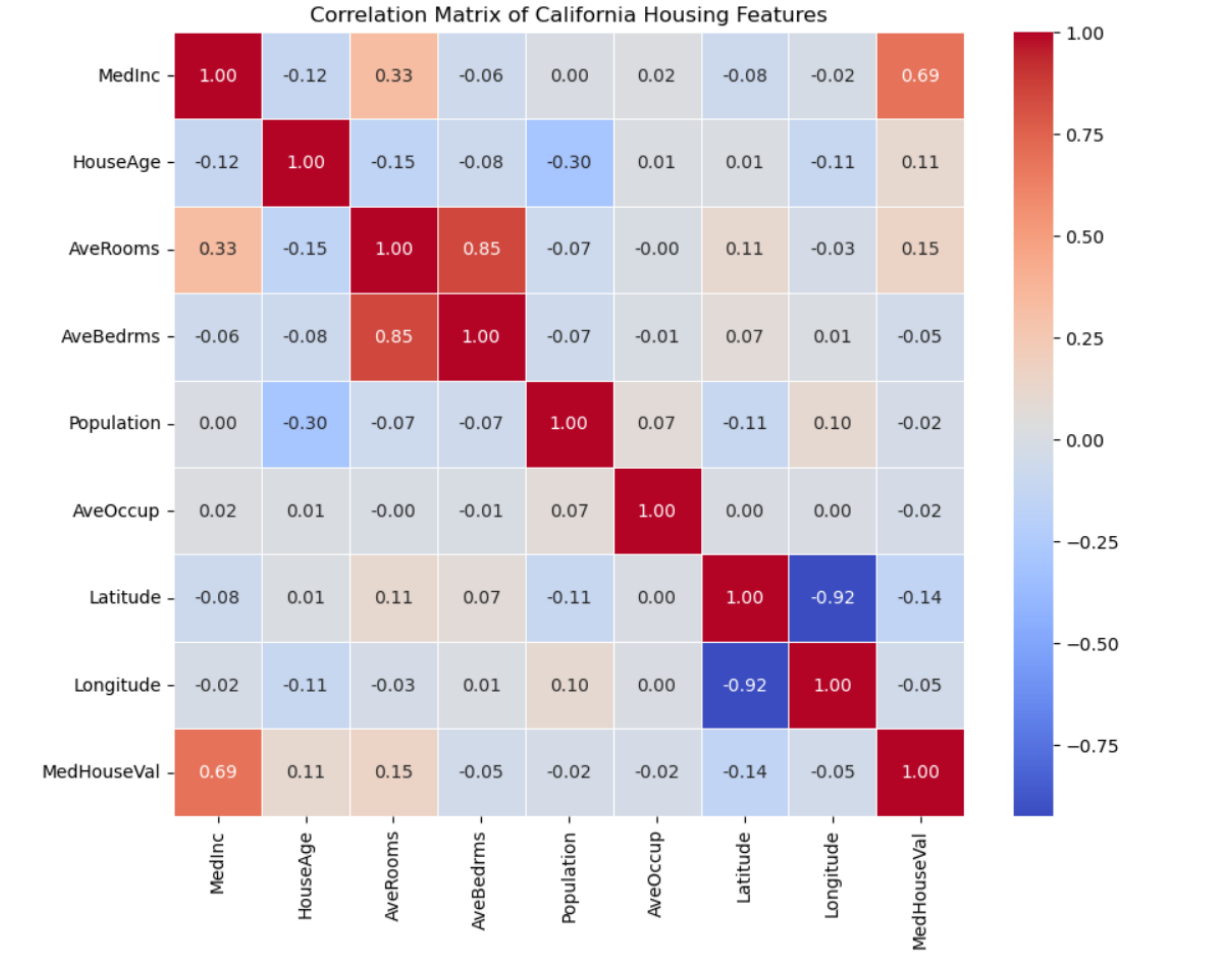
Develop a program to Compute the correlation matrix to understand the relationships between pairs of features. Visualize the correlation matrix using a heatmap to know which variables have strong positive/negative correlations. Create a pair plot to visualize pairwise relationships between features. Use California Housing dataset.

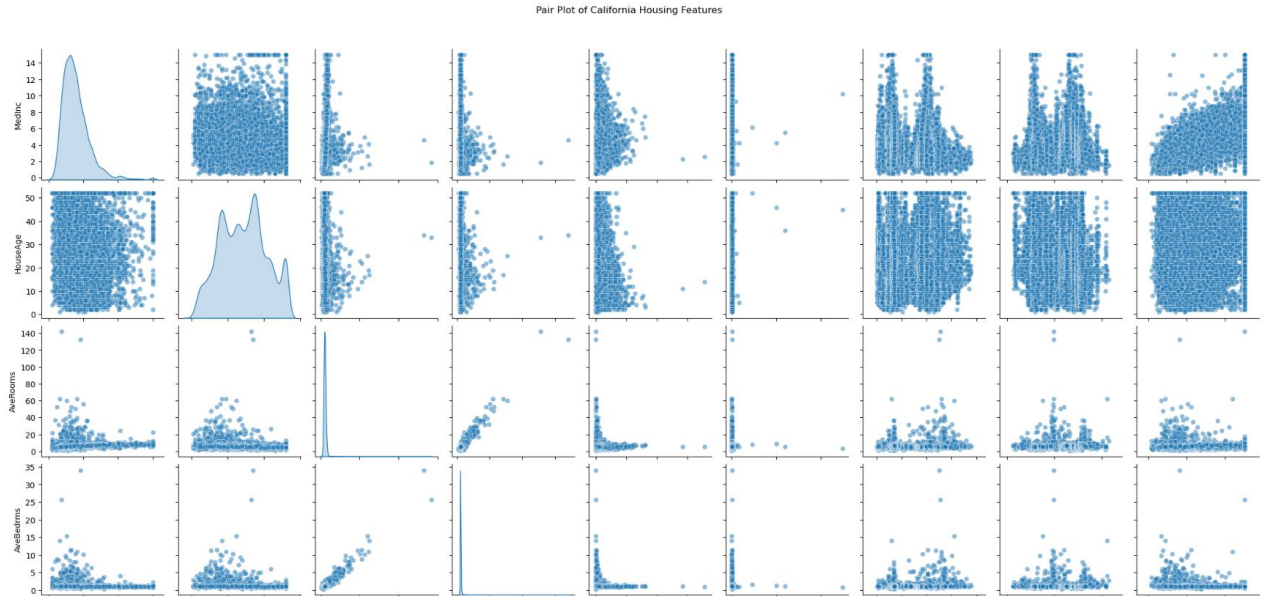
About the experiment: The program will be working with the **California Housing dataset** to explore the relationships between the features using **correlation analysis**. Correlation helps us understand how the variables are related to one another, which is essential for identifying key predictors and avoiding multicollinearity in machine learning models.

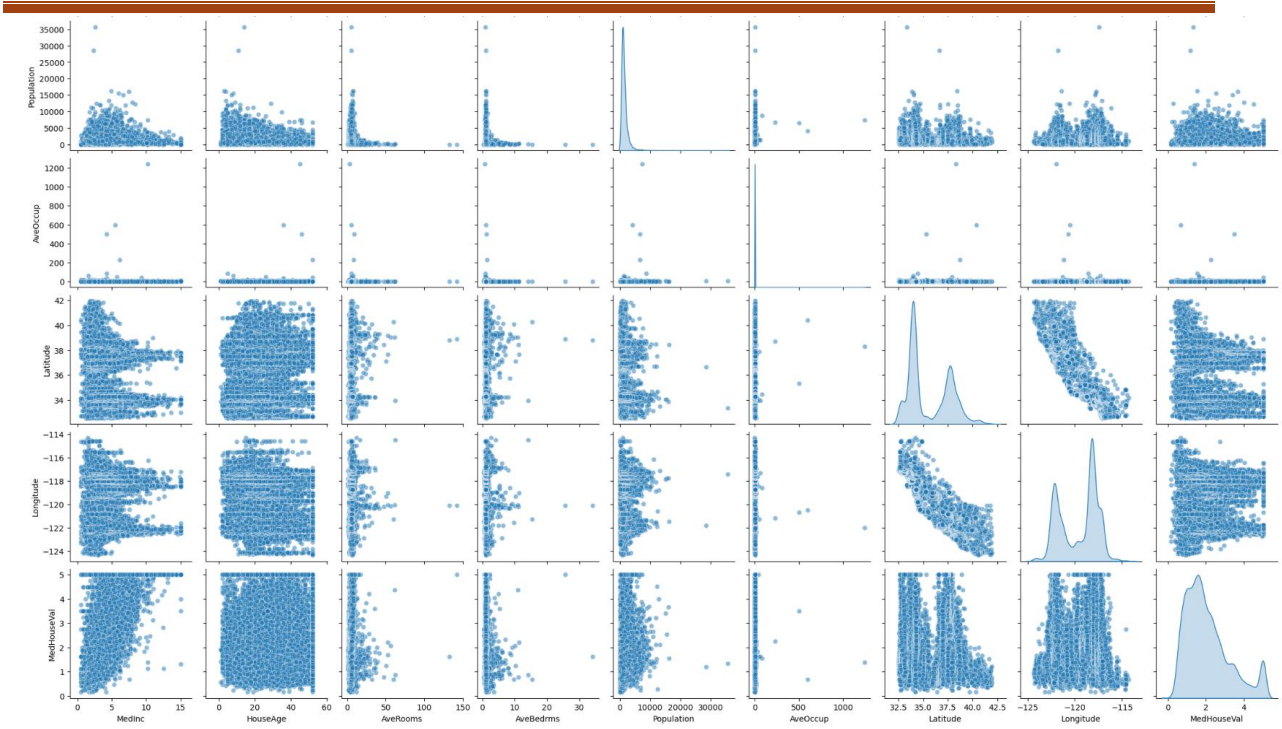
Program:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing

# Step 1: Load the California Housing Dataset
california_data = fetch_california_housing(as_frame=True)
data = california_data.frame

# Step 2: Compute the correlation matrix
correlation_matrix = data.corr()

# Step 3: Visualize the correlation matrix using a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Correlation Matrix of California Housing Features')
plt.show()

# Step 4: Create a pair plot to visualize pairwise relationships
sns.pairplot(data, diag_kind='kde', plot_kws={'alpha': 0.5})
plt.suptitle('Pair Plot of California Housing Features', y=1.02)
plt.show()
```

Output:


Correlation Matrix of California Housing Features
MedInc
HouseAge
AveRooms
AveBedrms
Population
AveOccup
Latitude
Longitude
MedHouseVal
1.00 -0.12 0.33 -0.06 0.00 0.02 -0.08 -0.02 0.69
-0.12 1.00 -0.15 -0.08 -0.30 0.01 0.01 -0.11 0.11
0.33 -0.15 1.00 0.85 -0.07 -0.00 0.11 -0.03 0.15
-0.06 -0.08 0.85 1.00 -0.07 -0.01 0.07 0.01 -0.05
0.00 -0.30 -0.07 -0.07 1.00 0.07 -0.11 0.10 -0.02
0.02 0.01 -0.00 -0.01 0.07 1.00 0.00 0.00 -0.02
-0.08 0.01 0.11 0.07 -0.11 0.00 1.00 -0.92 -0.14
-0.02 -0.11 -0.03 0.01 0.10 0.00 -0.92 1.00 -0.05
0.69 0.11 0.15 -0.05 -0.02 -0.02 -0.14 -0.05 1.00
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75


Pair Plot of California Housing Features
MedInc
HouseAge
AveRooms
AveBedrms


Population
AveOccup
Latitude
Longitude
MedHouseVal
MedInc
HouseAge
AveRooms
AveBedrms
Population
AveOccup
Latitude
Longitude
MedHouseVal

EXPERIMENT -3

Develop a program to implement Principal Component Analysis (PCA) for reducing the dimensionality of the Iris dataset from 4 features to 2.

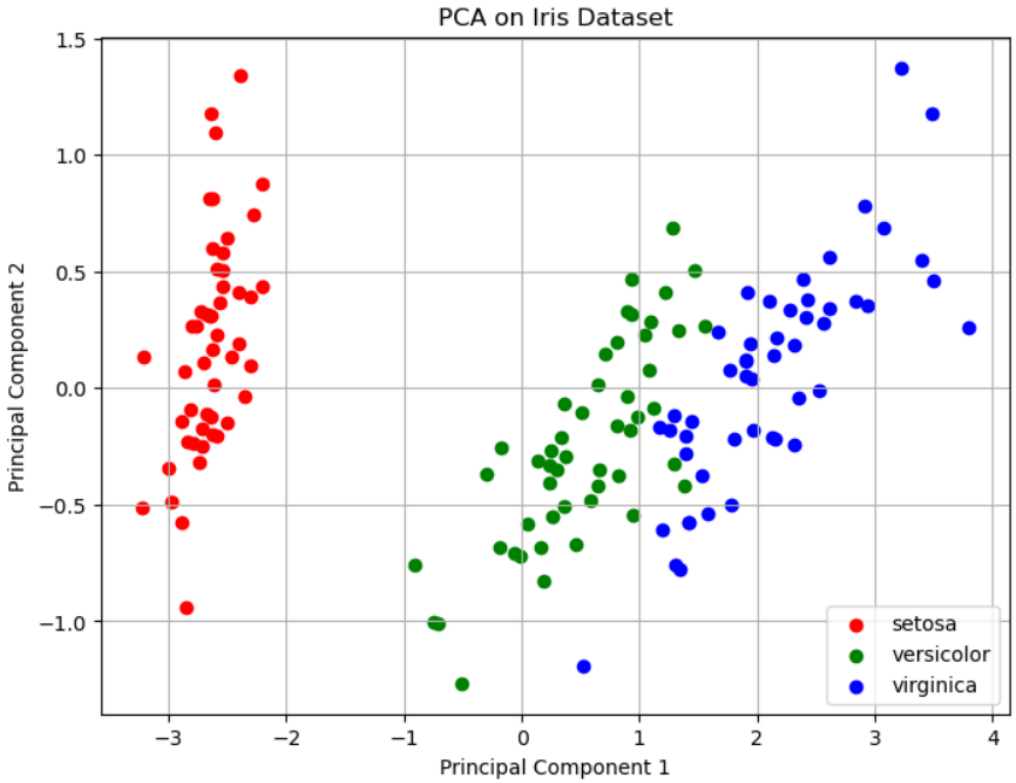
About the experiment: Dimensionality reduction is a crucial preprocessing step in machine learning and data visualization. Principal Component Analysis (PCA) is a popular technique used to transform high-dimensional data into a lower-dimensional space while retaining as much variance as possible. In this program, we will apply PCA to the Iris dataset, which consists of 4 features (sepal length, sepal width, petal length, and petal width). We aim to reduce these 4 dimensions to 2 while preserving the maximum amount of information. This will help in better visualization and analysis of the data.

Program:

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Load the Iris dataset
iris = load_iris()
data = iris.data
labels = iris.target
label_names = iris.target_names

# Convert to a DataFrame for better visualization
iris_df = pd.DataFrame(data, columns=iris.feature_names)

# Perform PCA to reduce dimensionality to 2
pca = PCA(n_components=2)
data_reduced = pca.fit_transform(data)

# Create a DataFrame for the reduced data
reduced_df = pd.DataFrame(data_reduced, columns=['Principal Component 1', 'Principal Component 2'])
reduced_df['Label'] = labels

# Plot the reduced data
plt.figure(figsize=(8, 6))
colors = ['r', 'g', 'b']
for i, label in enumerate(np.unique(labels)):
```

```
    plt.scatter(
        reduced_df[reduced_df['Label'] == label]['Principal Component 1'],
        reduced_df[reduced_df['Label'] == label]['Principal Component 2'],
        label=label_names[label],
        color=colors[i]
    )

plt.title('PCA on Iris Dataset')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.grid()
plt.show()
```

Output:



EXPERIMENT-4

For a given set of training data examples stored in a .CSV file, implement and demonstrate the Find-S algorithm to output a description of the set of all hypotheses consistent with the training examples.

About the experiment: The Find-S Algorithm is a simple, iterative algorithm used in concept learning to find the most specific hypothesis that is consistent with a given set of positive training examples. It starts with the most specific hypothesis and gradually generalizes it as it encounters more examples.

This program will read a CSV file containing training examples and apply the Find-S algorithm to learn the most specific hypothesis that fits the data. The dataset should have binary classification with attributes describing an object and a label indicating whether it belongs to the target concept (Yes/No or 1/0).

Program:

```
import pandas as pd

def find_s_algorithm(file_path):
    data = pd.read_csv(file_path)

    print("Training data:")
    print(data)

    attributes = data.columns[:-1]
    class_label = data.columns[-1]

    hypothesis = ['?' for _ in attributes]

    for index, row in data.iterrows():
        if row[class_label] == 'Yes':
            for i, value in enumerate(row[attributes]):
                if hypothesis[i] == '?' or hypothesis[i] == value:
                    hypothesis[i] = value
                else:
                    hypothesis[i] = '?'

    return hypothesis

file_path = 'training_data.csv'
```

```
hypothesis = find_s_algorithm(file_path)
print("\nThe final hypothesis is:", hypothesis)
```

Output:

```
Training data:
    Outlook Temperature Humidity  Windy PlayTennis
0     Sunny         Hot     High  False         No
1     Sunny         Hot     High   True         No
2  Overcast         Hot     High  False        Yes
3      Rain        Cold     High  False        Yes
4      Rain        Cold     High   True         No
5  Overcast         Hot     High   True        Yes
6     Sunny         Hot     High  False         No

The final hypothesis is: ['Overcast', 'Hot', 'High', '?']
```

EXPERIMENT 5

Develop a program to implement k-Nearest Neighbour algorithm to classify the randomly generated 100 values of *x* in the range of [0,1]. Perform the following based on dataset generated.

a. **Label the first 50 points {*x1*,……,*x50*} as follows: if (xi ≤ 0.5), then *xi* ε Class1, else *xi* ε Class1**
b. **Classify the remaining points, *x51*,……,*x100* using KNN. Perform this for *k=1,2,3,4,5,20,30***

About the experiment: The k-Nearest Neighbors (KNN) algorithm is a simple and effective classification technique that assigns a class to a new data point based on the majority class of its k nearest neighbors.

In this program, we will:

1. Generate 100 random values between 0 and 1.
2. Label the first 50 points such that:
   If xi≤0 assign Class 1
   Else, assign Class 2
3. Use KNN to classify the remaining 50 points ($x_{51},...,x_{100}$) for different values of k (1, 2, 3, 4, 5, 20, 30).

Program:

```
import numpy as np
import matplotlib.pyplot as plt
from collections import Counter

data = np.random.rand(100)
labels = ["Class1" if x <= 0.5 else "Class2" for x in data[:50]]


def euclidean_distance(x1, x2):
   return abs(x1 - x2)

def knn_classifier(train_data, train_labels, test_point, k):
        distances = [(euclidean_distance(test_point, train_data[i]), train_labels[i]) for i in
                                                                range(len(train_data))]

        distances.sort(key=lambda x: x[0])
        k_nearest_neighbors = distances[:k]

        k_nearest_labels = [label for _, label in k_nearest_neighbors]
```

```
        return Counter(k_nearest_labels).most_common(1)[0][0]


train_data = data[:50]
train_labels = labels

test_data = data[50:]

k_values = [1, 2, 3, 4, 5, 20, 30]

print("--- k-Nearest Neighbors Classification ---")
print("Training dataset: First 50 points labeled based on the rule (x <= 0.5 -> Class1, x > 0.5 -
> Class2)")
print("Testing dataset: Remaining 50 points to be classified\n")

results = {}

for k in k_values:
        print(f"Results for k = {k}:")
        classified_labels = [knn_classifier(train_data, train_labels, test_point, k) for test_point
                                                                                    in test_data]
        results[k] = classified_labels

        for i, label in enumerate(classified_labels, start=51):
                print(f"Point x{i} (value: {test_data[i - 51]:.4f}) is classified as {label}")
   print("\n")

print("Classification complete.\n")

for k in k_values:
            classified_labels = results[k]
            class1_points = [test_data[i] for i in range(len(test_data)) if classified_labels[i] ==
                                                                                    "Class1"]
            class2_points = [test_data[i] for i in range(len(test_data)) if classified_labels[i] ==
                                                                                    "Class2"]

            plt.figure(figsize=(10, 6))
            plt.scatter(train_data, [0] * len(train_data), c=["blue" if label == "Class1" else "red"
                                for label in train_labels], label="Training Data", marker="o")
            plt.scatter(class1_points, [1] * len(class1_points), c="blue", label="Class1 (Test)",
                                                                                    marker="x")
```

```
    plt.scatter(class2_points, [1] * len(class2_points), c="red", label="Class2 (Test)",
                                                                    marker="x")

    plt.title(f"k-NN Classification Results for k = {k}")
    plt.xlabel("Data Points")
    plt.ylabel("Classification Level")
    plt.legend()
    plt.grid(True)
    plt.show()
```

Output:

```
--- k-Nearest Neighbors Classification ---
Training dataset: First 50 points labeled based on the rule (x <= 0.5 -> Class1, x > 0.5
-> Class2)
Testing dataset: Remaining 50 points to be classified

Results for k = 1:
Point x51 (value: 0.3391) is classified as Class1
Point x52 (value: 0.1654) is classified as Class1
Point x53 (value: 0.5296) is classified as Class2
Point x54 (value: 0.5944) is classified as Class2
Point x55 (value: 0.4151) is classified as Class1
Point x56 (value: 0.2874) is classified as Class1
Point x57 (value: 0.4395) is classified as Class1
Point x58 (value: 0.0334) is classified as Class1
Point x59 (value: 0.0611) is classified as Class1
Point x60 (value: 0.6329) is classified as Class2
Point x61 (value: 0.6124) is classified as Class2
Point x62 (value: 0.2897) is classified as Class1
Point x63 (value: 0.3451) is classified as Class1
Point x64 (value: 0.1725) is classified as Class1
Point x65 (value: 0.1638) is classified as Class1
Point x66 (value: 0.0448) is classified as Class1
Point x67 (value: 0.2414) is classified as Class1
Point x68 (value: 0.7806) is classified as Class2
Point x69 (value: 0.4770) is classified as Class1
Point x70 (value: 0.9930) is classified as Class2
Point x71 (value: 0.7749) is classified as Class2
Point x72 (value: 0.6111) is classified as Class2
Point x73 (value: 0.3907) is classified as Class1
Point x74 (value: 0.6524) is classified as Class2
```

Point x75 (value: 0.2626) is classified as Class1
Point x76 (value: 0.3387) is classified as Class1
Point x77 (value: 0.8218) is classified as Class2
Point x78 (value: 0.3161) is classified as Class1
Point x79 (value: 0.6037) is classified as Class2
Point x80 (value: 0.9929) is classified as Class2
Point x81 (value: 0.2064) is classified as Class1
Point x82 (value: 0.1789) is classified as Class1
Point x83 (value: 0.3189) is classified as Class1
Point x84 (value: 0.6955) is classified as Class2
Point x85 (value: 0.1530) is classified as Class1
Point x86 (value: 0.1689) is classified as Class1
Point x87 (value: 0.5110) is classified as Class1
Point x88 (value: 0.2046) is classified as Class1
Point x89 (value: 0.4296) is classified as Class1
Point x90 (value: 0.5851) is classified as Class2
Point x91 (value: 0.0099) is classified as Class1
Point x92 (value: 0.9862) is classified as Class2
Point x93 (value: 0.3818) is classified as Class1
Point x94 (value: 0.0119) is classified as Class1
Point x95 (value: 0.9671) is classified as Class2
Point x96 (value: 0.1029) is classified as Class1
Point x97 (value: 0.5455) is classified as Class2
Point x98 (value: 0.2596) is classified as Class1
Point x99 (value: 0.3205) is classified as Class1
Point x100 (value: 0.9269) is classified as Class2

Results for k = 2:
Point x51 (value: 0.3391) is classified as Class1
Point x52 (value: 0.1654) is classified as Class1
Point x53 (value: 0.5296) is classified as Class2
Point x54 (value: 0.5944) is classified as Class2
Point x55 (value: 0.4151) is classified as Class1
Point x56 (value: 0.2874) is classified as Class1
Point x57 (value: 0.4395) is classified as Class1
Point x58 (value: 0.0334) is classified as Class1
Point x59 (value: 0.0611) is classified as Class1
Point x60 (value: 0.6329) is classified as Class2
Point x61 (value: 0.6124) is classified as Class2
Point x62 (value: 0.2897) is classified as Class1
Point x63 (value: 0.3451) is classified as Class1

Point x64 (value: 0.1725) is classified as Class1
Point x65 (value: 0.1638) is classified as Class1
Point x66 (value: 0.0448) is classified as Class1
Point x67 (value: 0.2414) is classified as Class1
Point x68 (value: 0.7806) is classified as Class2
Point x69 (value: 0.4770) is classified as Class1
Point x70 (value: 0.9930) is classified as Class2
Point x71 (value: 0.7749) is classified as Class2
Point x72 (value: 0.6111) is classified as Class2
Point x73 (value: 0.3907) is classified as Class1
Point x74 (value: 0.6524) is classified as Class2
Point x75 (value: 0.2626) is classified as Class1
Point x76 (value: 0.3387) is classified as Class1
Point x77 (value: 0.8218) is classified as Class2
Point x78 (value: 0.3161) is classified as Class1
Point x79 (value: 0.6037) is classified as Class2
Point x80 (value: 0.9929) is classified as Class2
Point x81 (value: 0.2064) is classified as Class1
Point x82 (value: 0.1789) is classified as Class1
Point x83 (value: 0.3189) is classified as Class1
Point x84 (value: 0.6955) is classified as Class2
Point x85 (value: 0.1530) is classified as Class1
Point x86 (value: 0.1689) is classified as Class1
Point x87 (value: 0.5110) is classified as Class1
Point x88 (value: 0.2046) is classified as Class1
Point x89 (value: 0.4296) is classified as Class1
Point x90 (value: 0.5851) is classified as Class2
Point x91 (value: 0.0099) is classified as Class1
Point x92 (value: 0.9862) is classified as Class2
Point x93 (value: 0.3818) is classified as Class1
Point x94 (value: 0.0119) is classified as Class1
Point x95 (value: 0.9671) is classified as Class2
Point x96 (value: 0.1029) is classified as Class1
Point x97 (value: 0.5455) is classified as Class2
Point x98 (value: 0.2596) is classified as Class1
Point x99 (value: 0.3205) is classified as Class1
Point x100 (value: 0.9269) is classified as Class2

Results for k = 3:
Point x51 (value: 0.3391) is classified as Class1
Point x52 (value: 0.1654) is classified as Class1

Point x53 (value: 0.5296) is classified as Class2
Point x54 (value: 0.5944) is classified as Class2
Point x55 (value: 0.4151) is classified as Class1
Point x56 (value: 0.2874) is classified as Class1
Point x57 (value: 0.4395) is classified as Class1
Point x58 (value: 0.0334) is classified as Class1
Point x59 (value: 0.0611) is classified as Class1
Point x60 (value: 0.6329) is classified as Class2
Point x61 (value: 0.6124) is classified as Class2
Point x62 (value: 0.2897) is classified as Class1
Point x63 (value: 0.3451) is classified as Class1
Point x64 (value: 0.1725) is classified as Class1
Point x65 (value: 0.1638) is classified as Class1
Point x66 (value: 0.0448) is classified as Class1
Point x67 (value: 0.2414) is classified as Class1
Point x68 (value: 0.7806) is classified as Class2
Point x69 (value: 0.4770) is classified as Class1
Point x70 (value: 0.9930) is classified as Class2
Point x71 (value: 0.7749) is classified as Class2
Point x72 (value: 0.6111) is classified as Class2
Point x73 (value: 0.3907) is classified as Class1
Point x74 (value: 0.6524) is classified as Class2
Point x75 (value: 0.2626) is classified as Class1
Point x76 (value: 0.3387) is classified as Class1
Point x77 (value: 0.8218) is classified as Class2
Point x78 (value: 0.3161) is classified as Class1
Point x79 (value: 0.6037) is classified as Class2
Point x80 (value: 0.9929) is classified as Class2
Point x81 (value: 0.2064) is classified as Class1
Point x82 (value: 0.1789) is classified as Class1
Point x83 (value: 0.3189) is classified as Class1
Point x84 (value: 0.6955) is classified as Class2
Point x85 (value: 0.1530) is classified as Class1
Point x86 (value: 0.1689) is classified as Class1
Point x87 (value: 0.5110) is classified as Class1
Point x88 (value: 0.2046) is classified as Class1
Point x89 (value: 0.4296) is classified as Class1
Point x90 (value: 0.5851) is classified as Class2
Point x91 (value: 0.0099) is classified as Class1
Point x92 (value: 0.9862) is classified as Class2
Point x93 (value: 0.3818) is classified as Class1
Point x94 (value: 0.0119) is classified as Class1

Point x95 (value: 0.9671) is classified as Class2
Point x96 (value: 0.1029) is classified as Class1
Point x97 (value: 0.5455) is classified as Class2
Point x98 (value: 0.2596) is classified as Class1
Point x99 (value: 0.3205) is classified as Class1
Point x100 (value: 0.9269) is classified as Class2

Results for k = 4:
Point x51 (value: 0.3391) is classified as Class1
Point x52 (value: 0.1654) is classified as Class1
Point x53 (value: 0.5296) is classified as Class2
Point x54 (value: 0.5944) is classified as Class2
Point x55 (value: 0.4151) is classified as Class1
Point x56 (value: 0.2874) is classified as Class1
Point x57 (value: 0.4395) is classified as Class1
Point x58 (value: 0.0334) is classified as Class1
Point x59 (value: 0.0611) is classified as Class1
Point x60 (value: 0.6329) is classified as Class2
Point x61 (value: 0.6124) is classified as Class2
Point x62 (value: 0.2897) is classified as Class1
Point x63 (value: 0.3451) is classified as Class1
Point x64 (value: 0.1725) is classified as Class1
Point x65 (value: 0.1638) is classified as Class1
Point x66 (value: 0.0448) is classified as Class1
Point x67 (value: 0.2414) is classified as Class1
Point x68 (value: 0.7806) is classified as Class2
Point x69 (value: 0.4770) is classified as Class1
Point x70 (value: 0.9930) is classified as Class2
Point x71 (value: 0.7749) is classified as Class2
Point x72 (value: 0.6111) is classified as Class2
Point x73 (value: 0.3907) is classified as Class1
Point x74 (value: 0.6524) is classified as Class2
Point x75 (value: 0.2626) is classified as Class1
Point x76 (value: 0.3387) is classified as Class1
Point x77 (value: 0.8218) is classified as Class2
Point x78 (value: 0.3161) is classified as Class1
Point x79 (value: 0.6037) is classified as Class2
Point x80 (value: 0.9929) is classified as Class2
Point x81 (value: 0.2064) is classified as Class1
Point x82 (value: 0.1789) is classified as Class1
Point x83 (value: 0.3189) is classified as Class1

Point x84 (value: 0.6955) is classified as Class2
Point x85 (value: 0.1530) is classified as Class1
Point x86 (value: 0.1689) is classified as Class1
Point x87 (value: 0.5110) is classified as Class1
Point x88 (value: 0.2046) is classified as Class1
Point x89 (value: 0.4296) is classified as Class1
Point x90 (value: 0.5851) is classified as Class2
Point x91 (value: 0.0099) is classified as Class1
Point x92 (value: 0.9862) is classified as Class2
Point x93 (value: 0.3818) is classified as Class1
Point x94 (value: 0.0119) is classified as Class1
Point x95 (value: 0.9671) is classified as Class2
Point x96 (value: 0.1029) is classified as Class1
Point x97 (value: 0.5455) is classified as Class2
Point x98 (value: 0.2596) is classified as Class1
Point x99 (value: 0.3205) is classified as Class1
Point x100 (value: 0.9269) is classified as Class2

Results for k = 5:
Point x51 (value: 0.3391) is classified as Class1
Point x52 (value: 0.1654) is classified as Class1
Point x53 (value: 0.5296) is classified as Class1
Point x54 (value: 0.5944) is classified as Class2
Point x55 (value: 0.4151) is classified as Class1
Point x56 (value: 0.2874) is classified as Class1
Point x57 (value: 0.4395) is classified as Class1
Point x58 (value: 0.0334) is classified as Class1
Point x59 (value: 0.0611) is classified as Class1
Point x60 (value: 0.6329) is classified as Class2
Point x61 (value: 0.6124) is classified as Class2
Point x62 (value: 0.2897) is classified as Class1
Point x63 (value: 0.3451) is classified as Class1
Point x64 (value: 0.1725) is classified as Class1
Point x65 (value: 0.1638) is classified as Class1
Point x66 (value: 0.0448) is classified as Class1
Point x67 (value: 0.2414) is classified as Class1
Point x68 (value: 0.7806) is classified as Class2
Point x69 (value: 0.4770) is classified as Class1
Point x70 (value: 0.9930) is classified as Class2
Point x71 (value: 0.7749) is classified as Class2
Point x72 (value: 0.6111) is classified as Class2

Point x73 (value: 0.3907) is classified as Class1
Point x74 (value: 0.6524) is classified as Class2
Point x75 (value: 0.2626) is classified as Class1
Point x76 (value: 0.3387) is classified as Class1
Point x77 (value: 0.8218) is classified as Class2
Point x78 (value: 0.3161) is classified as Class1
Point x79 (value: 0.6037) is classified as Class2
Point x80 (value: 0.9929) is classified as Class2
Point x81 (value: 0.2064) is classified as Class1
Point x82 (value: 0.1789) is classified as Class1
Point x83 (value: 0.3189) is classified as Class1
Point x84 (value: 0.6955) is classified as Class2
Point x85 (value: 0.1530) is classified as Class1
Point x86 (value: 0.1689) is classified as Class1
Point x87 (value: 0.5110) is classified as Class1
Point x88 (value: 0.2046) is classified as Class1
Point x89 (value: 0.4296) is classified as Class1
Point x90 (value: 0.5851) is classified as Class2
Point x91 (value: 0.0099) is classified as Class1
Point x92 (value: 0.9862) is classified as Class2
Point x93 (value: 0.3818) is classified as Class1
Point x94 (value: 0.0119) is classified as Class1
Point x95 (value: 0.9671) is classified as Class2
Point x96 (value: 0.1029) is classified as Class1
Point x97 (value: 0.5455) is classified as Class2
Point x98 (value: 0.2596) is classified as Class1
Point x99 (value: 0.3205) is classified as Class1
Point x100 (value: 0.9269) is classified as Class2

Results for k = 20:
Point x51 (value: 0.3391) is classified as Class1
Point x52 (value: 0.1654) is classified as Class1
Point x53 (value: 0.5296) is classified as Class1
Point x54 (value: 0.5944) is classified as Class2
Point x55 (value: 0.4151) is classified as Class1
Point x56 (value: 0.2874) is classified as Class1
Point x57 (value: 0.4395) is classified as Class1
Point x58 (value: 0.0334) is classified as Class1
Point x59 (value: 0.0611) is classified as Class1
Point x60 (value: 0.6329) is classified as Class2
Point x61 (value: 0.6124) is classified as Class2

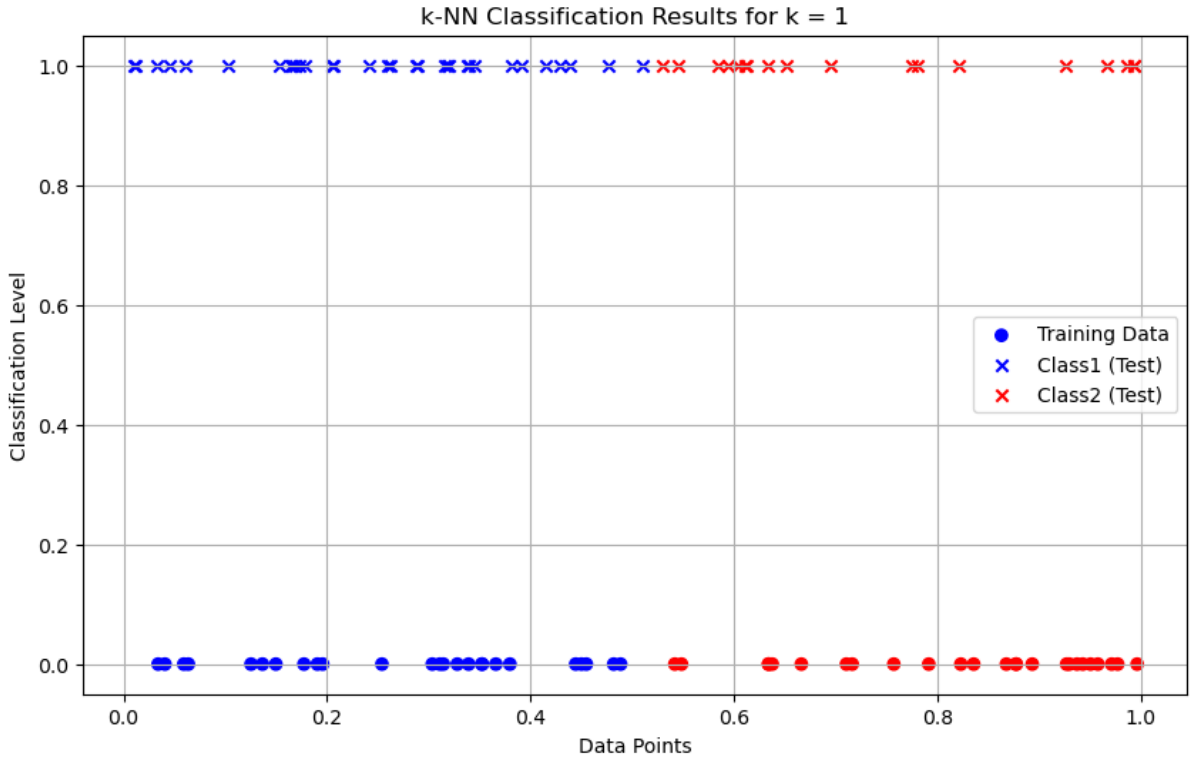
Point x62 (value: 0.2897) is classified as Class1
Point x63 (value: 0.3451) is classified as Class1
Point x64 (value: 0.1725) is classified as Class1
Point x65 (value: 0.1638) is classified as Class1
Point x66 (value: 0.0448) is classified as Class1
Point x67 (value: 0.2414) is classified as Class1
Point x68 (value: 0.7806) is classified as Class2
Point x69 (value: 0.4770) is classified as Class1
Point x70 (value: 0.9930) is classified as Class2
Point x71 (value: 0.7749) is classified as Class2
Point x72 (value: 0.6111) is classified as Class2
Point x73 (value: 0.3907) is classified as Class1
Point x74 (value: 0.6524) is classified as Class2
Point x75 (value: 0.2626) is classified as Class1
Point x76 (value: 0.3387) is classified as Class1
Point x77 (value: 0.8218) is classified as Class2
Point x78 (value: 0.3161) is classified as Class1
Point x79 (value: 0.6037) is classified as Class2
Point x80 (value: 0.9929) is classified as Class2
Point x81 (value: 0.2064) is classified as Class1
Point x82 (value: 0.1789) is classified as Class1
Point x83 (value: 0.3189) is classified as Class1
Point x84 (value: 0.6955) is classified as Class2
Point x85 (value: 0.1530) is classified as Class1
Point x86 (value: 0.1689) is classified as Class1
Point x87 (value: 0.5110) is classified as Class1
Point x88 (value: 0.2046) is classified as Class1
Point x89 (value: 0.4296) is classified as Class1
Point x90 (value: 0.5851) is classified as Class2
Point x91 (value: 0.0099) is classified as Class1
Point x92 (value: 0.9862) is classified as Class2
Point x93 (value: 0.3818) is classified as Class1
Point x94 (value: 0.0119) is classified as Class1
Point x95 (value: 0.9671) is classified as Class2
Point x96 (value: 0.1029) is classified as Class1
Point x97 (value: 0.5455) is classified as Class1
Point x98 (value: 0.2596) is classified as Class1
Point x99 (value: 0.3205) is classified as Class1
Point x100 (value: 0.9269) is classified as Class2

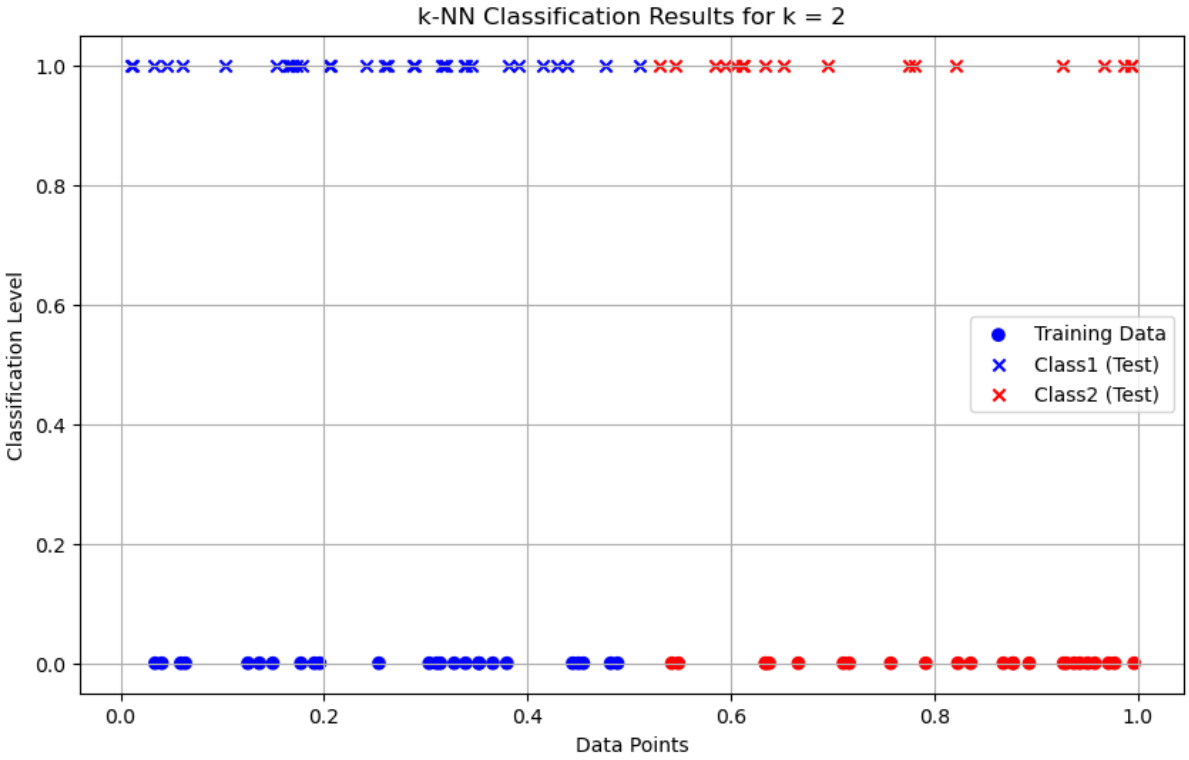
Results for k = 30:

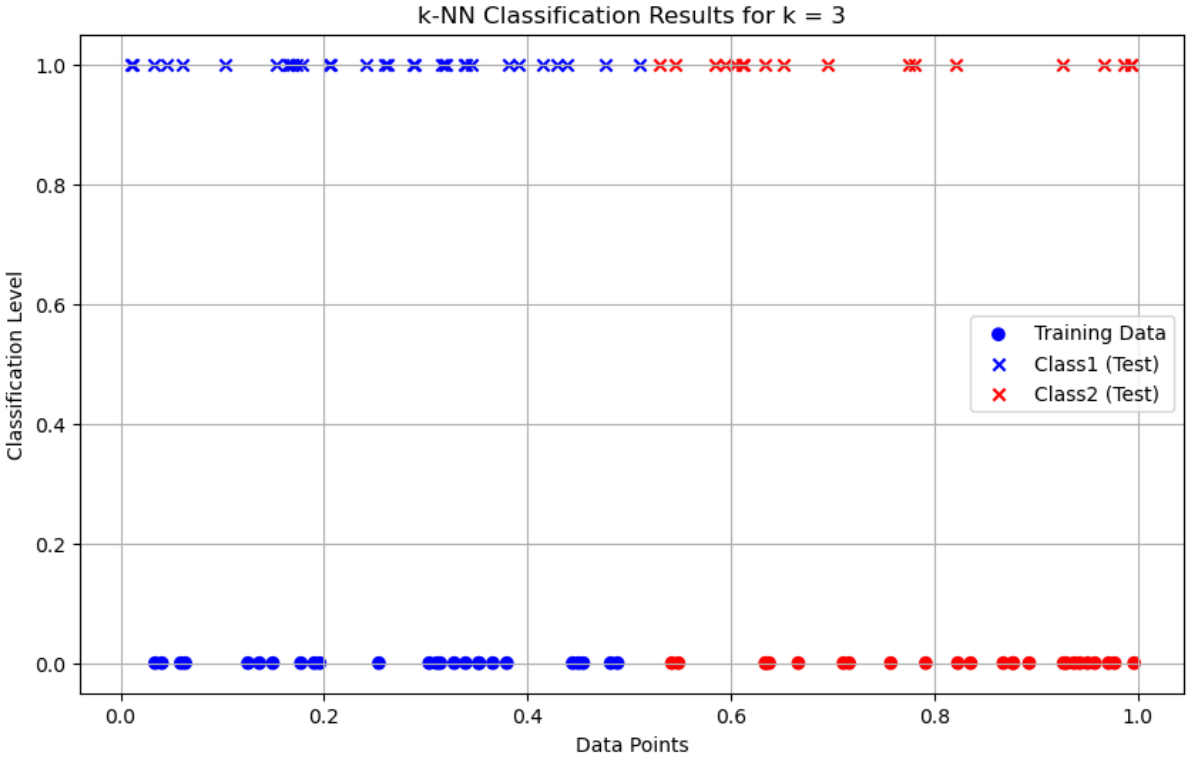
Point x51 (value: 0.3391) is classified as Class1
Point x52 (value: 0.1654) is classified as Class1
Point x53 (value: 0.5296) is classified as Class1
Point x54 (value: 0.5944) is classified as Class2
Point x55 (value: 0.4151) is classified as Class1
Point x56 (value: 0.2874) is classified as Class1
Point x57 (value: 0.4395) is classified as Class1
Point x58 (value: 0.0334) is classified as Class1
Point x59 (value: 0.0611) is classified as Class1
Point x60 (value: 0.6329) is classified as Class2
Point x61 (value: 0.6124) is classified as Class2
Point x62 (value: 0.2897) is classified as Class1
Point x63 (value: 0.3451) is classified as Class1
Point x64 (value: 0.1725) is classified as Class1
Point x65 (value: 0.1638) is classified as Class1
Point x66 (value: 0.0448) is classified as Class1
Point x67 (value: 0.2414) is classified as Class1
Point x68 (value: 0.7806) is classified as Class2
Point x69 (value: 0.4770) is classified as Class1
Point x70 (value: 0.9930) is classified as Class2
Point x71 (value: 0.7749) is classified as Class2
Point x72 (value: 0.6111) is classified as Class2
Point x73 (value: 0.3907) is classified as Class1
Point x74 (value: 0.6524) is classified as Class2
Point x75 (value: 0.2626) is classified as Class1
Point x76 (value: 0.3387) is classified as Class1
Point x77 (value: 0.8218) is classified as Class2
Point x78 (value: 0.3161) is classified as Class1
Point x79 (value: 0.6037) is classified as Class2
Point x80 (value: 0.9929) is classified as Class2
Point x81 (value: 0.2064) is classified as Class1
Point x82 (value: 0.1789) is classified as Class1
Point x83 (value: 0.3189) is classified as Class1
Point x84 (value: 0.6955) is classified as Class2
Point x85 (value: 0.1530) is classified as Class1
Point x86 (value: 0.1689) is classified as Class1
Point x87 (value: 0.5110) is classified as Class1
Point x88 (value: 0.2046) is classified as Class1
Point x89 (value: 0.4296) is classified as Class1
Point x90 (value: 0.5851) is classified as Class2
Point x91 (value: 0.0099) is classified as Class1
Point x92 (value: 0.9862) is classified as Class2

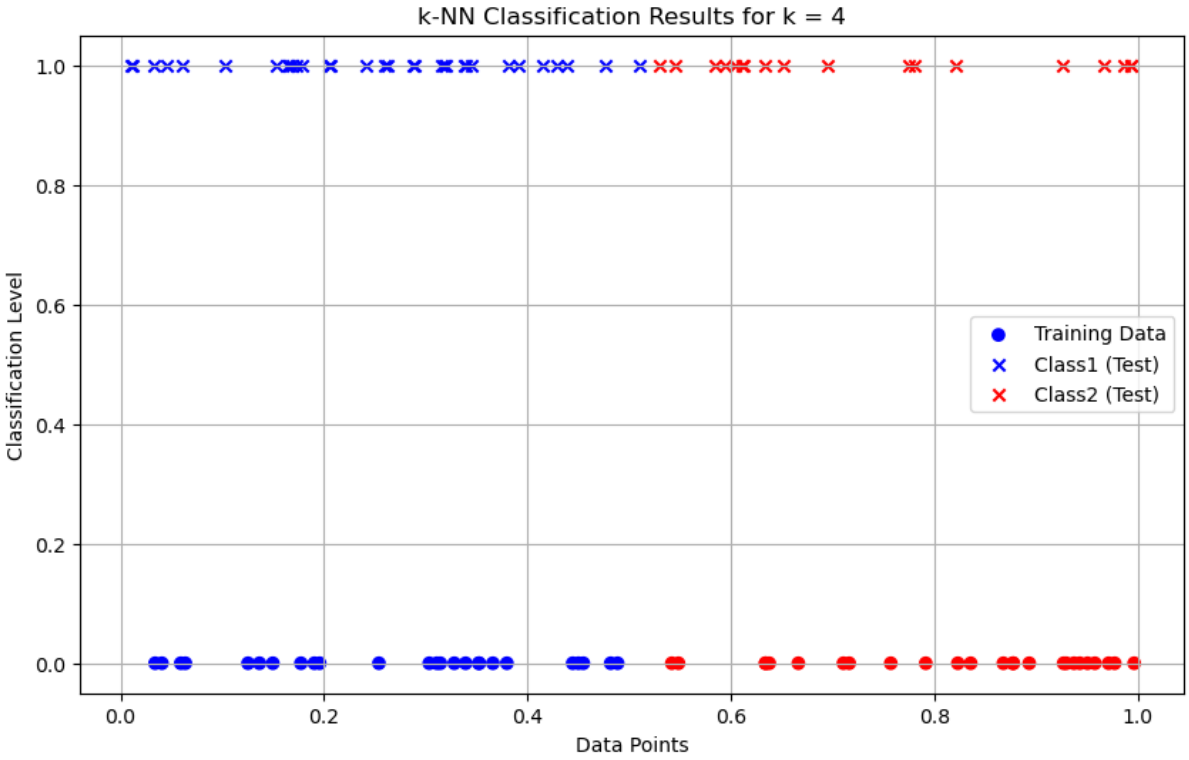
Point x93 (value: 0.3818) is classified as Class1
Point x94 (value: 0.0119) is classified as Class1
Point x95 (value: 0.9671) is classified as Class2
Point x96 (value: 0.1029) is classified as Class1
Point x97 (value: 0.5455) is classified as Class2
Point x98 (value: 0.2596) is classified as Class1
Point x99 (value: 0.3205) is classified as Class1
Point x100 (value: 0.9269) is classified as Class2

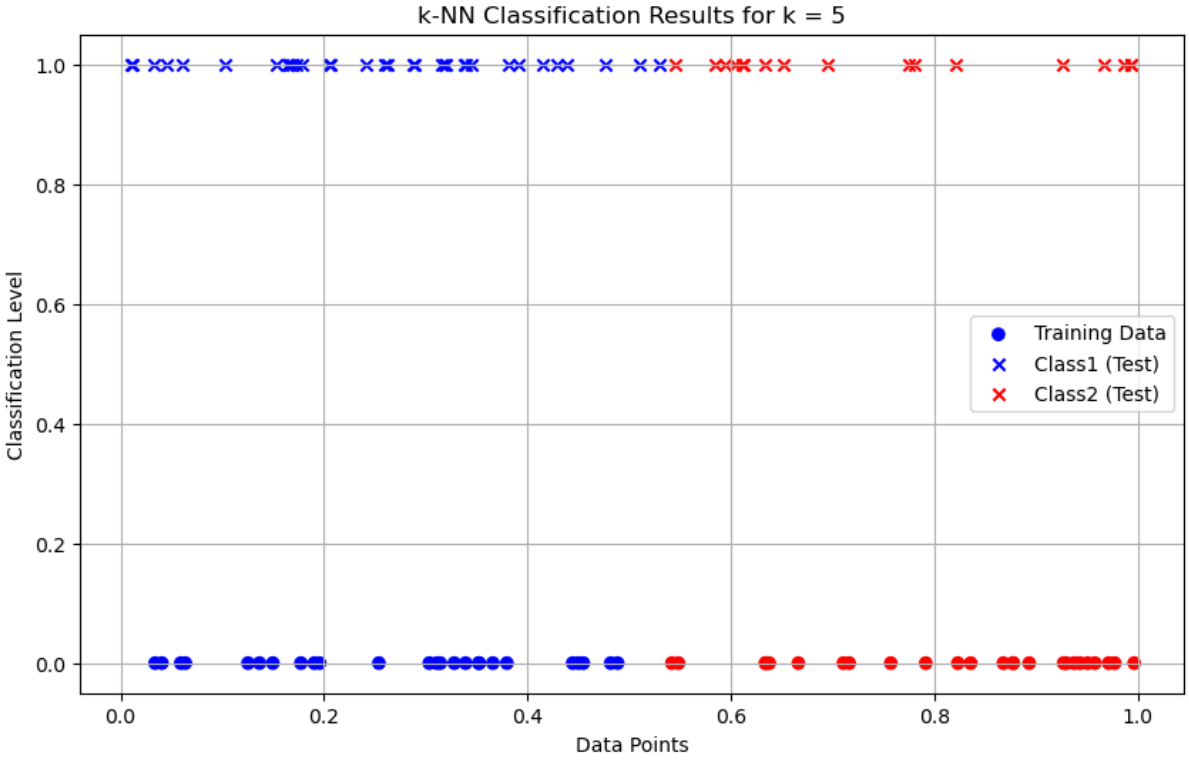
Classification complete.

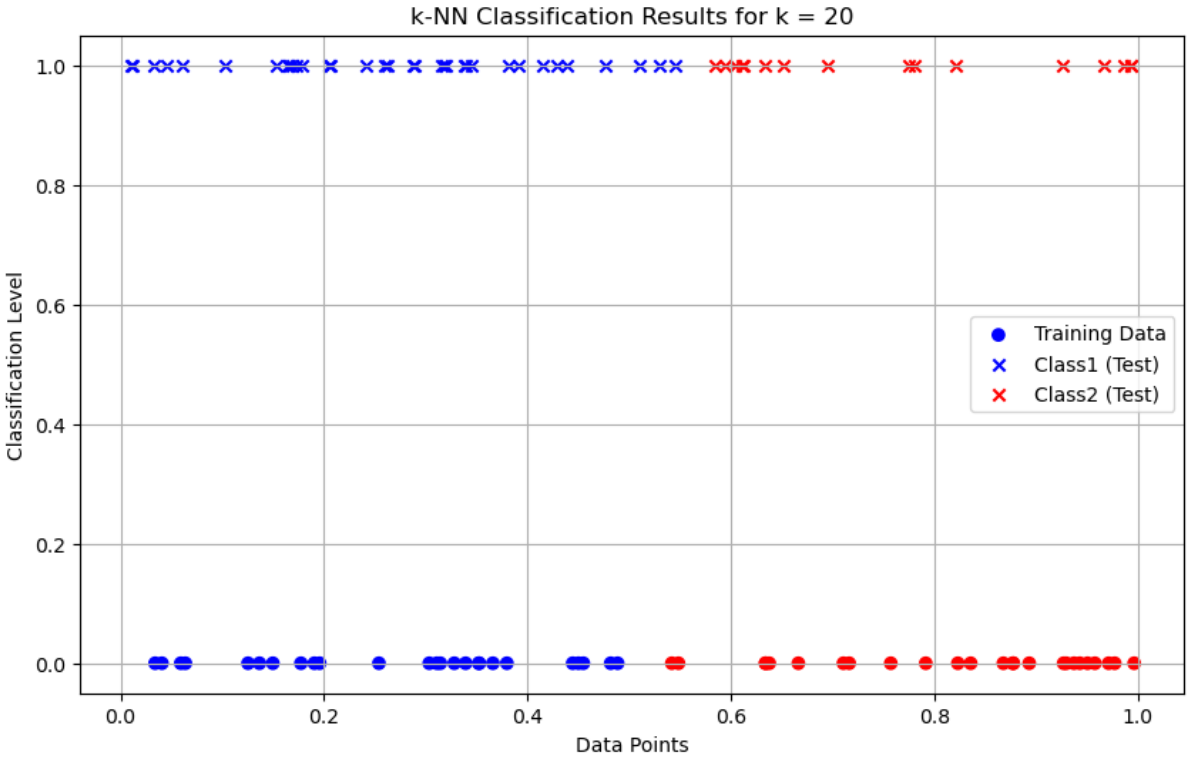


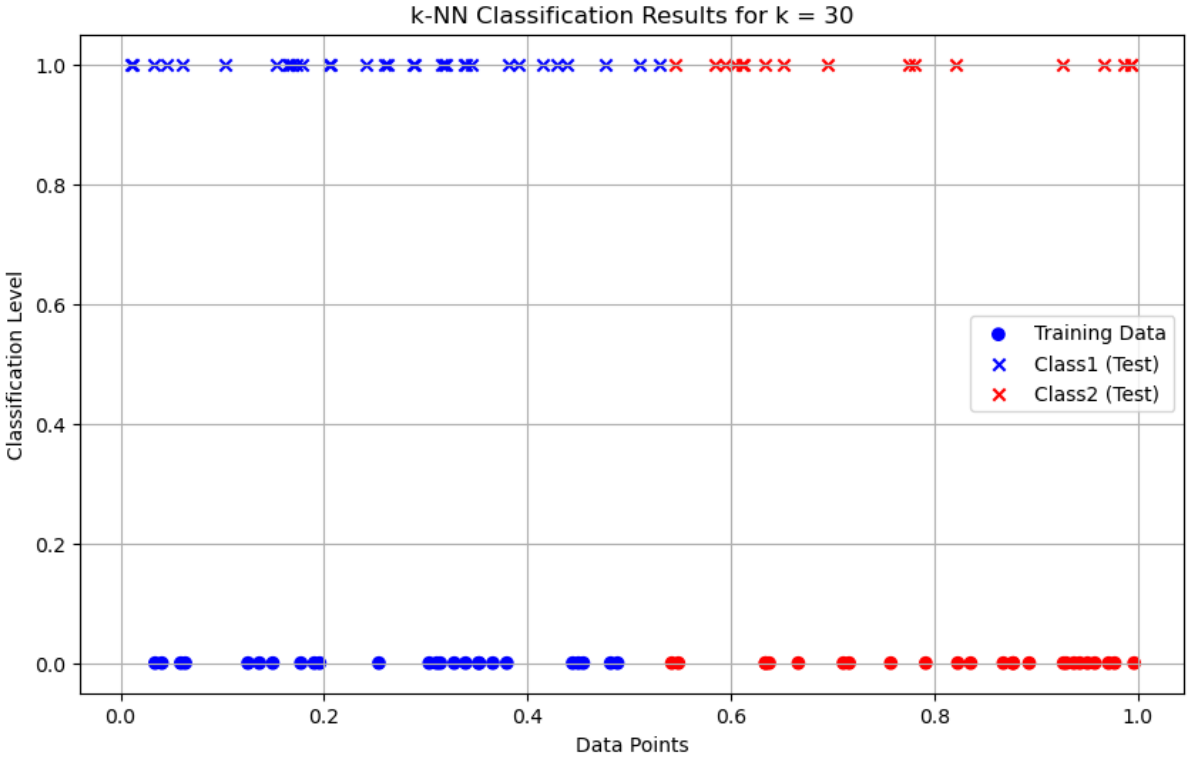

k-NN Classification Results for k = 2
Classification Level
Data Points
Training Data
Class1 (Test)
Class2 (Test)
0.0
0.2
0.4
0.6
0.8
1.0


k-NN Classification Results for k = 3
Classification Level
Data Points
Training Data
Class1 (Test)
Class2 (Test)
0.0
0.2
0.4
0.6
0.8
1.0


k-NN Classification Results for k = 4
Classification Level
Data Points
Training Data
Class1 (Test)
Class2 (Test)
0.0
0.2
0.4
0.6
0.8
1.0


k-NN Classification Results for k = 5
Classification Level
Data Points
Training Data
Class1 (Test)
Class2 (Test)
0.0
0.2
0.4
0.6
0.8
1.0


k-NN Classification Results for k = 20
Classification Level
Data Points
Training Data
Class1 (Test)
Class2 (Test)
0.0
0.2
0.4
0.6
0.8
1.0


k-NN Classification Results for k = 30
Classification Level
Data Points
Training Data
Class1 (Test)
Class2 (Test)
0.0
0.2
0.4
0.6
0.8
1.0

EXPERIMENT 6

Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.

About the experiment: Locally Weighted Regression (LWR), also known as Locally Weighted Scatterplot Smoothing (LOWESS/LOESS), is a non-parametric regression technique that fits a model to data points locally rather than applying a single global function. Unlike ordinary linear regression, which finds a single best-fit line, LWR assigns higher weights to nearby points and lower weights to distant points for each query point.
In this program, we will:

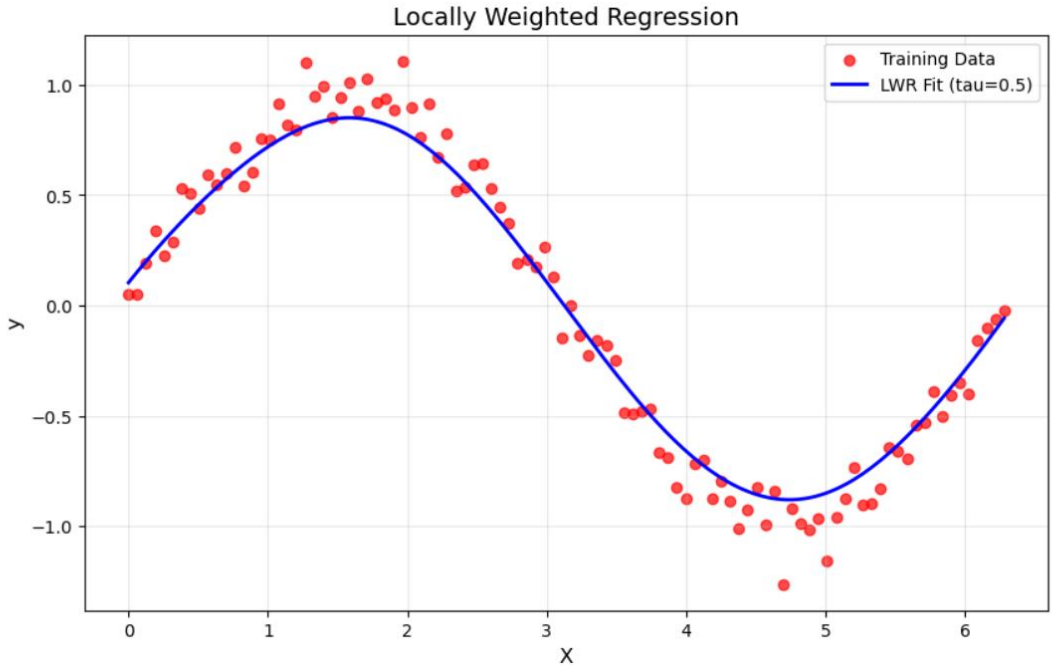
1. Generate a synthetic dataset.
2. Implement Locally Weighted Regression (LWR).
3. Fit the model to the data and visualize the smoothed curve.

Program:

```
import numpy as np
import matplotlib.pyplot as plt


def gaussian_kernel(x, xi, tau):
    return np.exp(-np.sum((x - xi) ** 2) / (2 * tau ** 2))

def locally_weighted_regression(x, X, y, tau):
    m = X.shape[0]
    weights = np.array([gaussian_kernel(x, X[i], tau) for i in range(m)])
    W = np.diag(weights)
    X_transpose_W = X.T @ W
    theta = np.linalg.inv(X_transpose_W @ X) @ X_transpose_W @ y
    return x @ theta

np.random.seed(42)
X = np.linspace(0, 2 * np.pi, 100)
y = np.sin(X) + 0.1 * np.random.randn(100)
X_bias = np.c_[np.ones(X.shape), X]

x_test = np.linspace(0, 2 * np.pi, 200)
x_test_bias = np.c_[np.ones(x_test.shape), x_test]
tau = 0.5
y_pred = np.array([locally_weighted_regression(xi, X_bias, y, tau) for xi in x_test_bias])

plt.figure(figsize=(10, 6))
```

```
plt.scatter(X, y, color='red', label='Training Data', alpha=0.7)
plt.plot(x_test, y_pred, color='blue', label=f'LWR Fit (tau={tau})', linewidth=2)
plt.xlabel('X', fontsize=12)
plt.ylabel('y', fontsize=12)
plt.title('Locally Weighted Regression', fontsize=14)
plt.legend(fontsize=10)
plt.grid(alpha=0.3)
plt.show()
```

Output:



EXPERIMENT 7

Develop a program to demonstrate the working of Linear Regression and Polynomial Regression. Use Boston Housing Dataset for Linear Regression and Auto MPG Dataset (for vehicle fuel efficiency prediction) for Polynomial Regression.

About the experiment: Regression analysis is a fundamental technique in machine learning used to model the relationship between dependent and independent variables. In this program, we will implement and demonstrate the working of Linear Regression and Polynomial Regression using two well-known datasets:

1. Linear Regression with the Boston Housing Dataset
   The Boston Housing Dataset contains information about various factors affecting housing prices in Boston, such as crime rate, number of rooms, and distance to employment centers. Using Linear Regression, we aim to predict the median house prices based on these features.
2. Polynomial Regression with the Auto MPG Dataset
   The Auto MPG Dataset provides information about automobile fuel efficiency (miles per gallon, MPG) based on attributes such as engine displacement, horsepower, and weight. Since the relationship between MPG and these features is non-linear, we use Polynomial Regression to capture the complex dependencies and improve prediction accuracy.

This program will cover:

- Data preprocessing and visualization
- Model training and evaluation for both regression techniques
- Comparison of Linear and Polynomial Regression performances

Program:

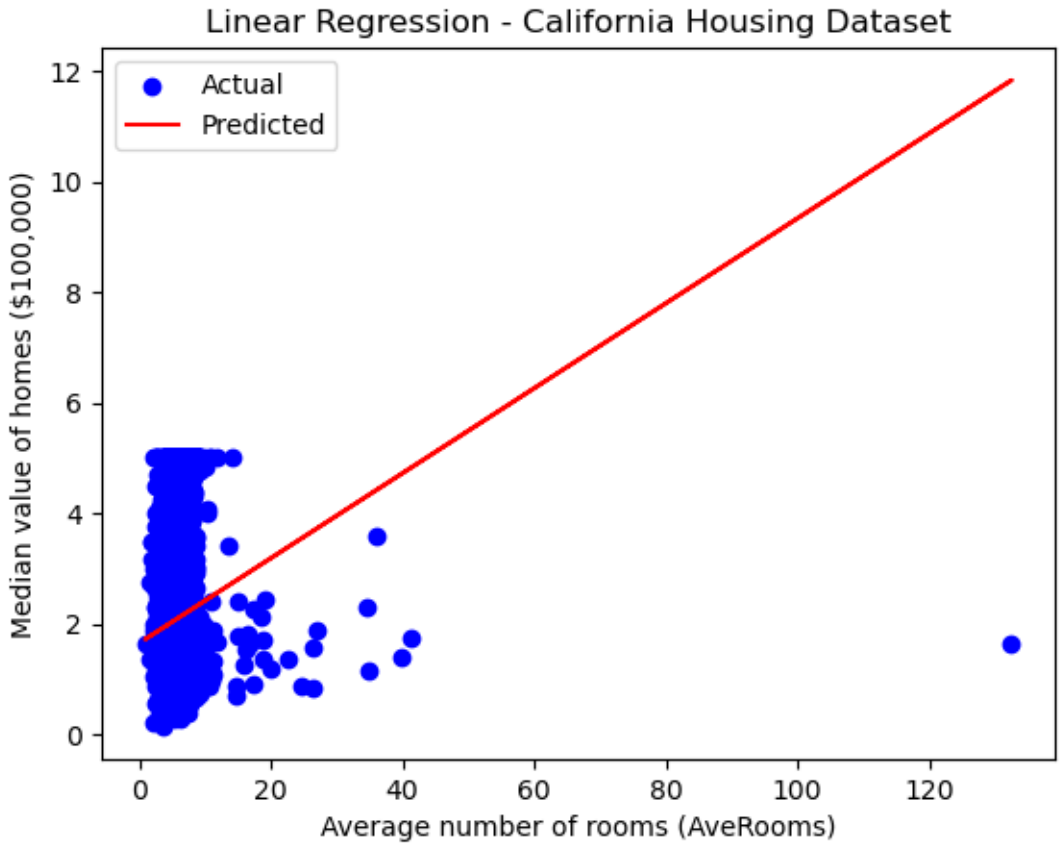
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.metrics import mean_squared_error, r2_score

def linear_regression_california():
    housing = fetch_california_housing(as_frame=True)
    X = housing.data[["AveRooms"]]
    y = housing.target
```

```
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    model = LinearRegression()
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)

    plt.scatter(X_test, y_test, color="blue", label="Actual")
    plt.plot(X_test, y_pred, color="red", label="Predicted")
    plt.xlabel("Average number of rooms (AveRooms)")
    plt.ylabel("Median value of homes ($100,000)")
    plt.title("Linear Regression - California Housing Dataset")
    plt.legend()
    plt.show()

    print("Linear Regression - California Housing Dataset")
    print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
    print("R^2 Score:", r2_score(y_test, y_pred))


def polynomial_regression_auto_mpg():
    url = "https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data"
    column_names = ["mpg", "cylinders", "displacement", "horsepower", "weight",
"acceleration", "model_year", "origin"]
    data = pd.read_csv(url, sep='\s+', names=column_names, na_values="?")
    data = data.dropna()

    X = data["displacement"].values.reshape(-1, 1)
    y = data["mpg"].values

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    poly_model = make_pipeline(PolynomialFeatures(degree=2), StandardScaler(),
LinearRegression())
    poly_model.fit(X_train, y_train)

    y_pred = poly_model.predict(X_test)

    plt.scatter(X_test, y_test, color="blue", label="Actual")
    plt.scatter(X_test, y_pred, color="red", label="Predicted")
    plt.xlabel("Displacement")
    plt.ylabel("Miles per gallon (mpg)")
    plt.title("Polynomial Regression - Auto MPG Dataset")
    plt.legend()
```

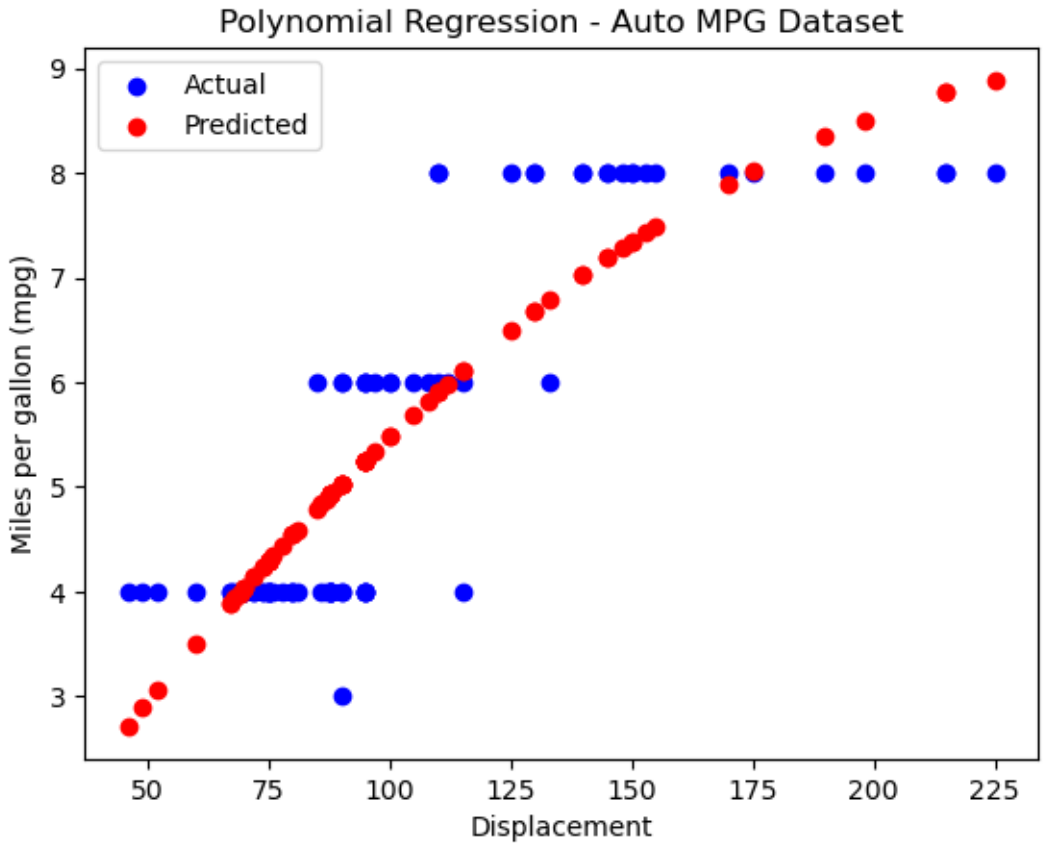
```
    plt.show()

    print("Polynomial Regression - Auto MPG Dataset")
    print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
    print("R^2 Score:", r2_score(y_test, y_pred))


if __name__ == "__main__":
    print("Demonstrating Linear Regression and Polynomial Regression\n")
    linear_regression_california()
    polynomial_regression_auto_mpg()
```

Output:

Demonstrating Linear Regression and Polynomial Regression



Linear Regression - California Housing Dataset
Mean Squared Error: 1.2923314440807299
R^2 Score: 0.013795337532284901



Polynomial Regression - Auto MPG Dataset
Mean Squared Error: 0.7431490557205862
R^2 Score: 0.7505650609469626

EXPERIMENT 8

Develop a program to demonstrate the working of the decision tree algorithm. Use Breast Cancer Data set for building the decision tree and apply this knowledge to classify a new sample.

About the Experiment: Introduction to Decision Tree Algorithm using Breast Cancer Dataset Decision Trees is a fundamental machine learning algorithm used for classification and regression tasks. They work by recursively splitting data based on feature values to create a tree-like structure that makes predictions.
In this program, we will demonstrate the working of a Decision Tree Classifier using the Breast Cancer Dataset from scikit-learn. This dataset contains various features of cell nuclei extracted from breast cancer biopsy samples and is used to classify tumors as malignant (cancerous) or benign (non-cancerous).
The program will:

- Load and explore the Breast Cancer dataset
- Train a Decision Tree Classifier
- Evaluate the model's performance
- Classify a new sample based on the trained model

Program:

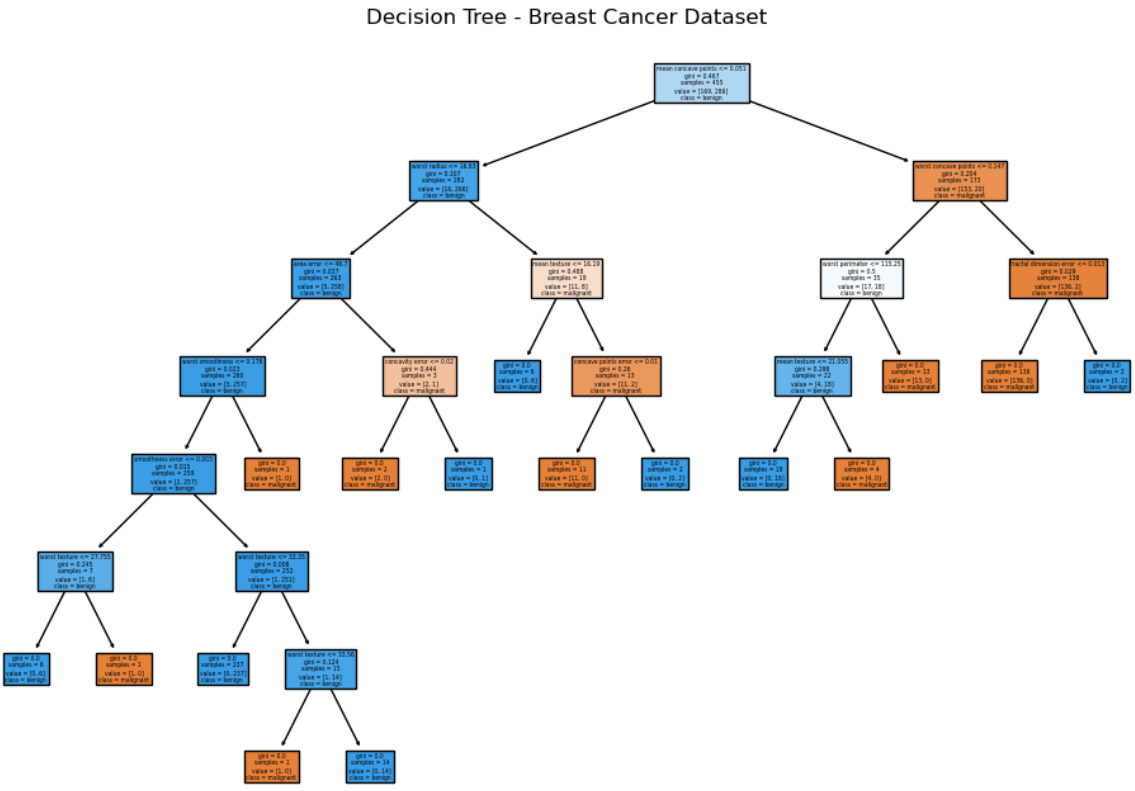
```
# Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree

data = load_breast_cancer()
X = data.data
y = data.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy * 100:.2f}%")
new_sample = np.array([X_test[0]])
```

```
prediction = clf.predict(new_sample)

prediction_class = "Benign" if prediction == 1 else "Malignant"
print(f"Predicted Class for the new sample: {prediction_class}")

plt.figure(figsize=(12,8))
tree.plot_tree(clf,filled=True,feature_names=data.feature_names,
                    class_names=data. target_names)
plt.title("Decision Tree - Breast Cancer Dataset")
 plt.show()
```

Output:

Model Accuracy: 94.74%

Predicted Class for the new sample: Benign



EXPERIMENT 9

Develop a program to implement the Naive Bayesian classifier considering Olivetti Face Data set for training. Compute the accuracy of the classifier, considering a few test data sets.

About the Experiment: The Naïve Bayes Classifier is a probabilistic machine learning algorithm based on Bayes' Theorem, assuming that features are conditionally independent given the class label. It is widely used for classification tasks such as text classification, spam detection, and facial recognition.

In this program, we will use the Olivetti Faces dataset, a well-known dataset for face recognition, to train a Naïve Bayes classifier and evaluate its accuracy. The dataset consists of 400 grayscale face images (64x64 pixels) of 40 different individuals. Each individual has 10 images, making it a great dataset for face classification.

Program:

```
import numpy as np
from sklearn.datasets import fetch_olivetti_faces
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt

data = fetch_olivetti_faces(shuffle=True, random_state=42)
X = data.data
y = data.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')

print("\nClassification Report:")
print(classification_report(y_test, y_pred, zero_division=1))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
cross_val_accuracy = cross_val_score(gnb, X, y, cv=5, scoring='accuracy')
print(f'\nCross-validation accuracy: {cross_val_accuracy.mean() * 100:.2f}%')

fig, axes = plt.subplots(3, 5, figsize=(12, 8))
for ax, image, label, prediction in zip(axes.ravel(), X_test, y_test, y_pred):
    ax.imshow(image.reshape(64, 64), cmap=plt.cm.gray)
    ax.set_title(f"True: {label}, Pred: {prediction}")
    ax.axis('off')

plt.show()
```

Output:

downloading Olivetti faces from https://ndownloader.figshare.com/files/5976027 to C:\Users\Divya U H\scikit_learn_data

Accuracy: 80.83%

Classification Report:

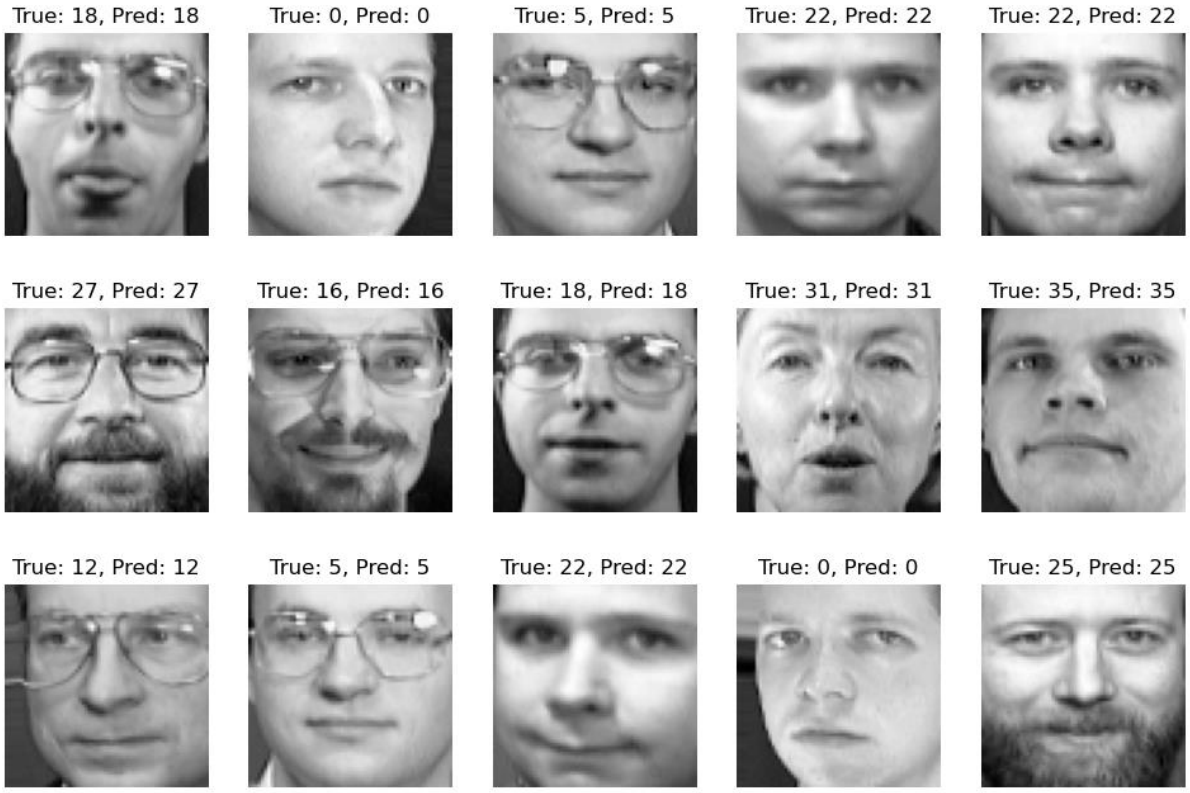
	precision	recall	f1-score	support
0	0.67	1.00	0.80	2
1	1.00	1.00	1.00	2
2	0.33	0.67	0.44	3
3	1.00	0.00	0.00	5
4	1.00	0.50	0.67	4
5	1.00	1.00	1.00	2
7	1.00	0.75	0.86	4
8	1.00	0.67	0.80	3
9	1.00	0.75	0.86	4
10	1.00	1.00	1.00	3
11	1.00	1.00	1.00	1
12	0.40	1.00	0.57	4
13	1.00	0.80	0.89	5
14	1.00	0.40	0.57	5
15	0.67	1.00	0.80	2
16	1.00	0.67	0.80	3
17	1.00	1.00	1.00	3
18	1.00	1.00	1.00	3
19	0.67	1.00	0.80	2
20	1.00	1.00	1.00	3
21	1.00	0.67	0.80	3
22	1.00	0.60	0.75	5
23	1.00	0.75	0.86	4
24	1.00	1.00	1.00	3

25	1.00	0.75	0.86	4
26	1.00	1.00	1.00	2
27	1.00	1.00	1.00	5
28	0.50	1.00	0.67	2
29	1.00	1.00	1.00	2
30	1.00	1.00	1.00	2
31	1.00	0.75	0.86	4
32	1.00	1.00	1.00	2
34	0.25	1.00	0.40	1
35	1.00	1.00	1.00	5
36	1.00	1.00	1.00	3
37	1.00	1.00	1.00	1
38	1.00	0.75	0.86	4
39	0.50	1.00	0.67	5
accuracy			0.81	120
macro avg	0.89	0.85	0.83	120
weighted avg	0.91	0.81	0.81	120

Confusion Matrix:

```
[[2 0 0 ... 0 0 0]
 [0 2 0 ... 0 0 0]
 [0 0 2 ... 0 0 1]
 ...
 [0 0 0 ... 1 0 0]
 [0 0 0 ... 0 3 0]
 [0 0 0 ... 0 0 5]]
```

Cross-validation accuracy: 87.25%


True: 18, Pred: 18
True: 0, Pred: 0
True: 5, Pred: 5
True: 22, Pred: 22
True: 22, Pred: 22
True: 27, Pred: 27
True: 16, Pred: 16
True: 18, Pred: 18
True: 31, Pred: 31
True: 35, Pred: 35
True: 12, Pred: 12
True: 5, Pred: 5
True: 22, Pred: 22
True: 0, Pred: 0
True: 25, Pred: 25

EXPERIMENT 10

Develop a program to implement k-means clustering using Wisconsin Breast Cancer data set and visualize the clustering result.

About the Experiment: K-Means clustering is an unsupervised learning algorithm used to group similar data points into clusters. The Wisconsin Breast Cancer dataset contains features of tumors, and we will apply K-Means to group the data into two clusters (malignant and benign tumors).

Objective:

1. Apply K-Means clustering to categorize tumors into two clusters.
2. Use Principal Component Analysis (PCA) to reduce dimensions for visualization.
3. Evaluate clustering performance by comparing with actual labels.
4. Visualize the clusters to understand the results.

Program:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_breast_cancer
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.metrics import confusion_matrix, classification_report

# Load the Breast Cancer dataset
data = load_breast_cancer()
X = data.data  # Features
y = data.target  # True labels (0 = Malignant, 1 = Benign)

# Standardize the data (K-Means works better with normalized data)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply K-Means clustering (n_clusters=2 for two types of tumors)
kmeans = KMeans(n_clusters=2, random_state=42, n_init=10)  # Fix warning by setting n_init=10
y_kmeans = kmeans.fit_predict(X_scaled)

# Print evaluation metrics
print("Confusion Matrix:")
print(confusion_matrix(y, y_kmeans))
```

```
print("\nClassification Report:")
print(classification_report(y, y_kmeans))

# Reduce dimensions using PCA for visualization (2D)
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Convert to DataFrame for easy visualization
df = pd.DataFrame(X_pca, columns=['PC1', 'PC2'])
df['Cluster'] = y_kmeans  # Clustering result
df['True Label'] = y  # Actual labels

# Scatter plot of clusters
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='PC1', y='PC2', hue='Cluster', palette='Set1', s=100,
edgecolor='black', alpha=0.7)
plt.title('K-Means Clustering of Breast Cancer Dataset')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title="Cluster")
plt.show()

# Scatter plot of true labels
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='PC1', y='PC2', hue='True Label', palette='coolwarm', s=100,
edgecolor='black', alpha=0.7)
plt.title('True Labels of Breast Cancer Dataset')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title="True Label")
plt.show()

# Visualizing cluster centers
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='PC1', y='PC2', hue='Cluster', palette='Set1', s=100,
edgecolor='black', alpha=0.7)
centers = pca.transform(kmeans.cluster_centers_)  # Transform cluster centers to PCA space
plt.scatter(centers[:, 0], centers[:, 1], s=200, c='red', marker='X', label='Centroids')
plt.title('K-Means Clustering with Centroids')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title="Cluster")
plt.show()
```

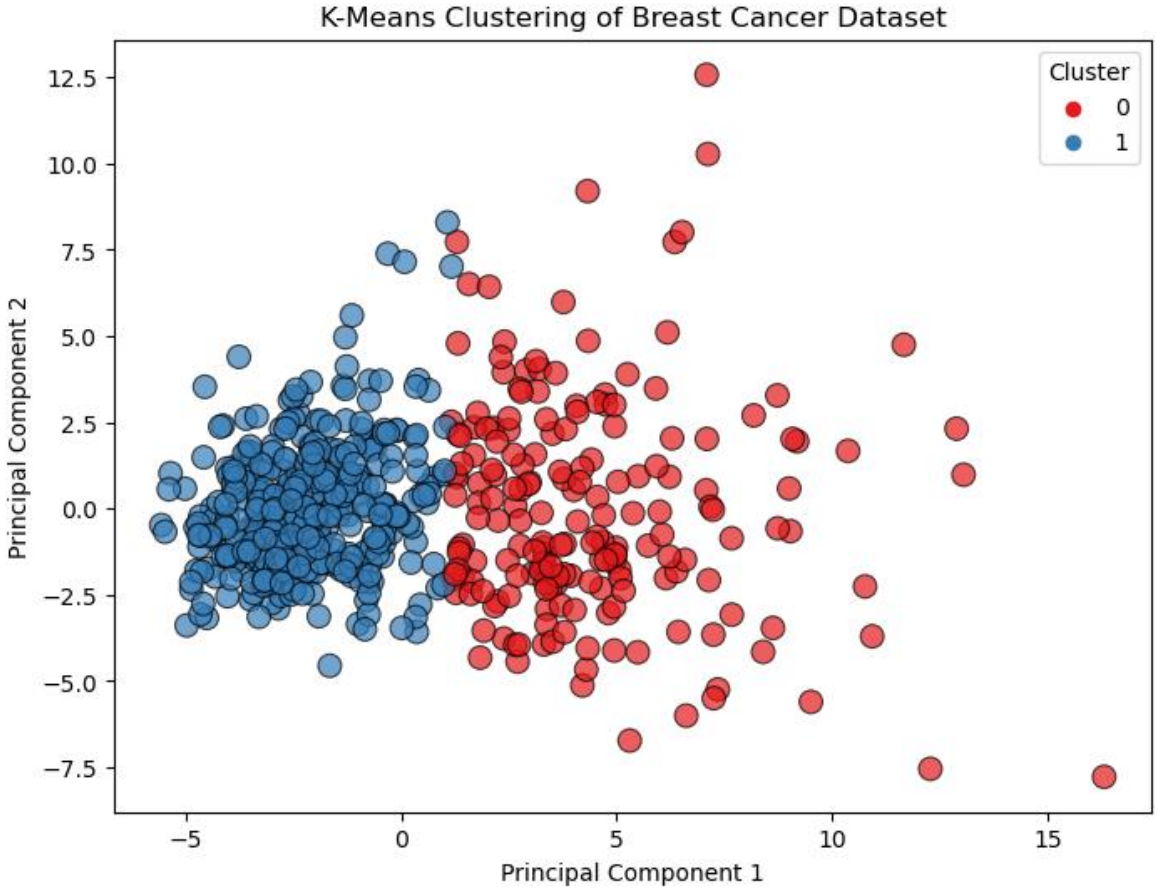
Output:

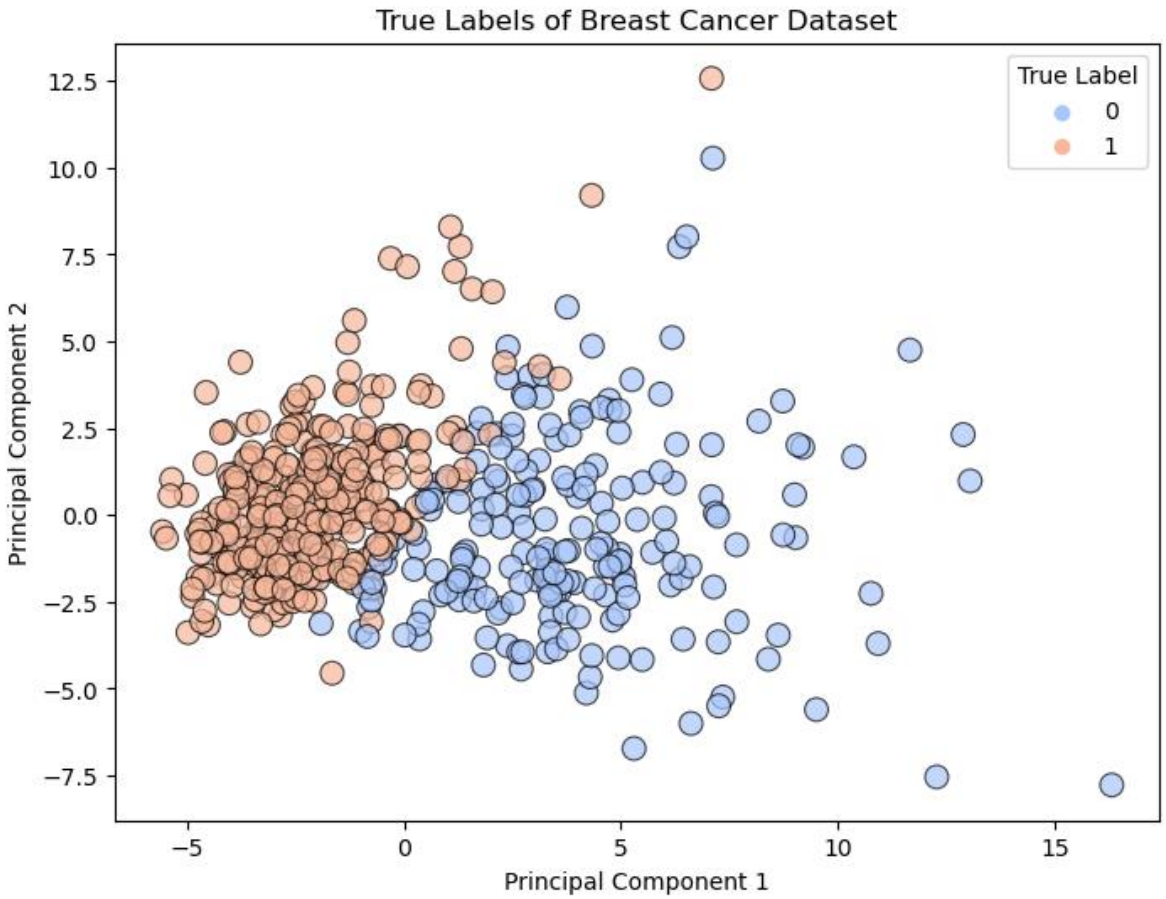
Confusion Matrix:

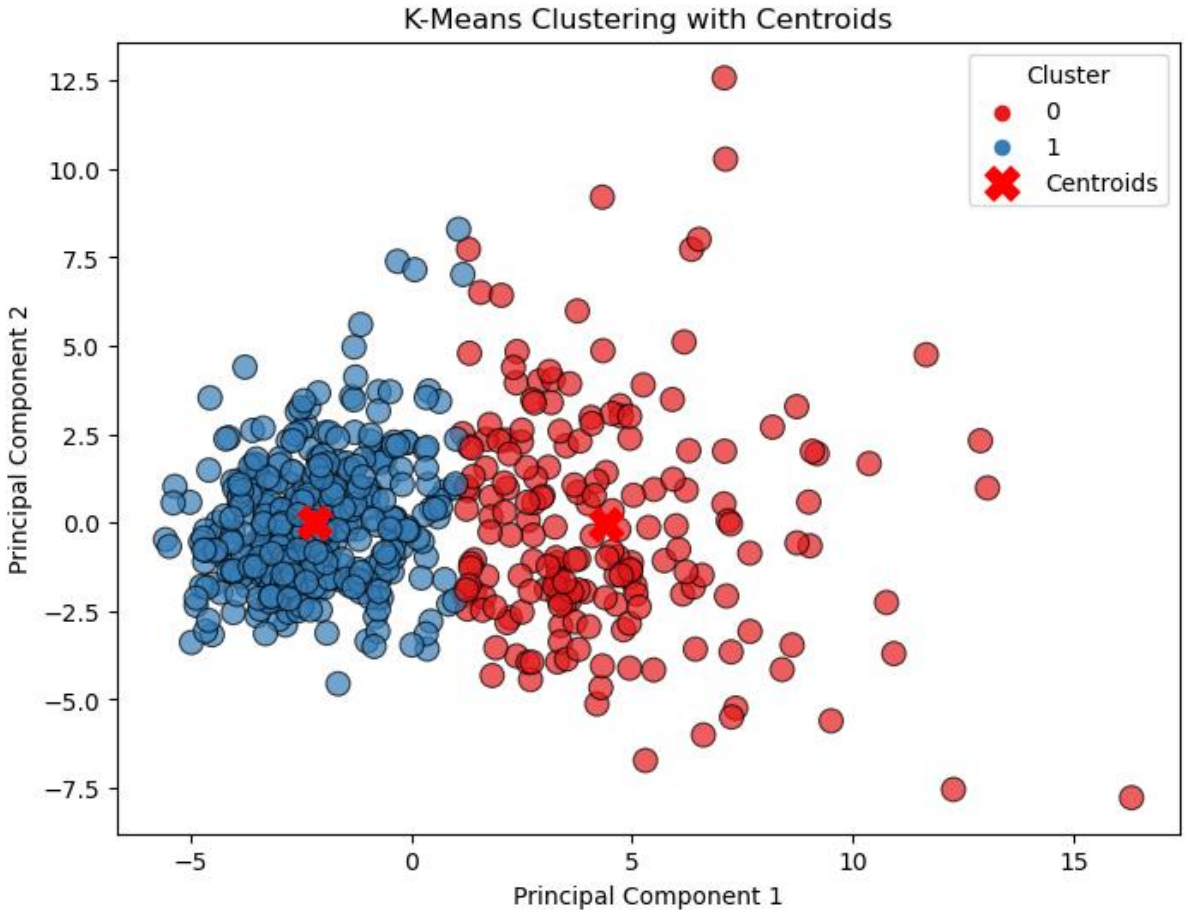
```
[[175  37]
 [ 14 343]]
```

Classification Report:

	precision	recall	f1-score	support
0	0.93	0.83	0.87	212
1	0.90	0.96	0.93	357
accuracy			0.91	569
macro avg	0.91	0.89	0.90	569
weighted avg	0.91	0.91	0.91	569




True Labels of Breast Cancer Dataset
True Label
0
1
Principal Component 2
Principal Component 1
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−5
0
5
10
15


K-Means Clustering with Centroids
Cluster
0
1
Centroids
Principal Component 2
Principal Component 1
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−5
0
5
10
15

Viva-Voce Questions

Q1: What is the purpose of a histogram?
A: A histogram shows the distribution of a numerical variable by dividing data into bins.

Q2: What do box plots help identify?
A: Box plots help identify the median, quartiles, and outliers in the dataset.

Q3: What is an outlier?
A: An outlier is a data point that lies significantly outside the range of the other observations.

Q4: What does the correlation coefficient measure?
A: It measures the strength and direction of the linear relationship between two variables.

Q5: What is the range of the correlation coefficient?
A: The correlation coefficient ranges from -1 to +1.

Q6: What does a high positive correlation mean?
A: A high positive correlation means that as one variable increases, the other also increases.

Q7: What is the main goal of PCA?
A: PCA reduces the dimensionality of a dataset while preserving as much variance as possible.

Q8: How does PCA achieve dimensionality reduction?
A: PCA transforms data into new axes (principal components) that maximize variance.

Q9: What is the first principal component?
A: The first principal component is the direction that captures the most variance in the data.

Q10: What is the Find-S algorithm used for?
A: It is used to find the most specific hypothesis that fits all positive examples.

Q11: What type of learning is Find-S?
A: It is an **inductive learning** algorithm in **supervised learning**.

Q12: Why does Find-S ignore negative examples?
A: Find-S only generalizes from positive examples to create a specific hypothesis.

Q13: What is the k-NN algorithm used for?
A: k-NN is a **classification** and **regression** algorithm based on proximity to training data points.

Q14: What happens when k=1 in k-NN?
A: The nearest neighbor determines the class of the test sample.

Q15: What is the effect of increasing k in k-NN?
A: A larger k smooths the decision boundary, reducing overfitting.

Q16: What is Locally Weighted Regression (LWR)?
A: LWR is a non-parametric regression algorithm that fits local models around each point.

Q17: How does LWR differ from ordinary linear regression?
A: LWR assigns weights to nearby points instead of using a global model.

Q18: Why is LWR computationally expensive?
A: LWR recalculates weights and fits a new model for each query point.

Q19: What is the difference between linear and polynomial regression?
A: Linear regression fits a straight line, whereas **polynomial regression** fits a curve.

Q20: Why do we use polynomial regression?
A: To model non-linear relationships between the independent and dependent variables.

Q21: What dataset is used for linear regression in the program?
A: Boston Housing Dataset.

Q22: What is a decision tree?
A: A **supervised learning** model that makes decisions by splitting data into branches.

Q23: What is the criterion used for splitting in a decision tree?
A: It can be **Gini impurity** or **Entropy (Information Gain)**.

Q24: What is overfitting in decision trees?
A: Overfitting occurs when the tree learns noise instead of the pattern in the training data.

Q25: Why is the Naive Bayes classifier called "naive"?
A: It assumes that all features are independent given the class label.

Q26: What is the probability model used in Naive Bayes?
A: It uses **Bayes' theorem** to compute conditional probabilities.

Q27: What assumption is made in the Gaussian Naive Bayes model?
A: It assumes that features follow a **normal (Gaussian) distribution**.

Q28: What is the objective of k-means clustering?
A: To partition the dataset into **k clusters** by minimizing the distance within each cluster.

Q29: How does k-means decide the cluster assignment?
A: By assigning each point to the nearest cluster **centroid**.

Q30: What metric is used to evaluate clustering performance?
A: Confusion matrix (if true labels are available) or **Silhouette Score**.

****